

Db2 11.1 for Linux, UNIX, and Windows



Built-In Routines and Views

Db2 11.1 for Linux, UNIX, and Windows



Built-In Routines and Views

Notice regarding this document

This document in PDF form is provided as a courtesy to customers who have requested documentation in this format. It is provided As-Is without warranty or maintenance commitment.

Contents

Notice regarding this document iii

Figures ix

Tables xi

Built-in routines and views 1

Best practices for calling built-in routines and views in applications. 1

Authorizations for using built-in routines and views . 2

Built-in views versus table functions 3

Supported built-in SQL routines and views 3

 Administrative routines and ADMIN_CMD procedure 4

 Administrative Task Scheduler routines and views 290

 Audit routines and procedures 303

 Automatic maintenance routines 306

 Common SQL API procedures. 312

 Configuration routines and views 363

 Db2 pureScale instance information routines and views 371

 Environment routines and views 381

 Explain routines 402

 Monitor views 421

 Monitor procedures and functions 459

 WebSphere MQ routines. 904

 Security routines and views 924

 Snapshot routines and views 938

 SQL procedure routines 989

 Stepwise redistribute routines 1000

 Storage management tool routines 1009

 Text search routines 1013

 Workload management routines. 1053

 Miscellaneous routines and views 1075

Deprecated SQL administrative routines and views. 1119

 ADMIN_GET_DBP_MEM_USAGE table function - Get total memory consumption for instance 1124

 ADMIN_TABCOMPRESSINFO administrative view and ADMIN_GET_TAB_COMPRESS_INFO table function (deprecated) - returns compressed information. 1126

 ADMIN_GET_TAB_COMPRESS_INFO_V97 1132

 ADMIN_GET_TAB_INFO_V95 table function - Retrieve size and state information for tables . 1138

 ADMIN_GET_TAB_INFO_V97 table function - Retrieve size and state information for tables . 1143

 AM_BASE_RPT_RECOMS - Recommendations for activity reports 1150

 AM_BASE_RPTS - Activity event monitor reports 1152

AM_DROP_TASK - Delete a monitoring task 1153

AM_GET_LOCK_CHN_TB - Retrieve application lock chain data in a tabular format . 1154

AM_GET_LOCK_CHNS - Retrieve lock chain information for a specific application 1155

AM_GET_LOCK_RPT - Retrieve application lock details 1156

AM_GET_RPT - Retrieve activity monitor data . . 1163

AM_SAVE_TASK - Create or modify a monitoring task 1164

APPL_PERFORMANCE administrative view - Retrieve percentage of rows selected for an application 1165

APPLICATIONS administrative view - Retrieve connected database application information . . 1167

APPLICATION_ID 1171

BP_HITRATIO administrative view - Retrieve bufferpool hit ratio information 1172

BP_READ_IO administrative view - Retrieve bufferpool read performance information. . . 1174

BP_WRITE_IO administrative view - Retrieve bufferpool write performance information . . 1176

CONTAINER_UTILIZATION administrative view - Retrieve table space container and utilization information 1178

DB_PARTITIONS. 1180

ENV_SYS_RESOURCES administrative view - Return system information 1182

ENV_GET_SYS_RESOURCES table function - return system information. 1185

GET_DB_CONFIG 1188

GET_DBM_CONFIG. 1190

Health snapshot routines 1191

LOCKS_HELD administrative view - Retrieve information about locks held 1236

LOCKWAITS administrative view - Retrieve current lockwaits information 1239

LOG_UTILIZATION administrative view - Retrieve log utilization information. 1242

LONG_RUNNING_SQL administrative view 1244

REG_VARIABLES administrative view - Retrieve database registry settings in use. . . 1246

REG_LIST_VARIABLES table function - return registry settings 1248

SNAPAGENT administrative view and SNAP_GET_AGENT table function - Retrieve agent logical data group application snapshot information 1250

SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function - Retrieve memory_pool logical data group snapshot information 1254

SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function - Retrieve appl_info logical data group snapshot information	1258
SNAP_GET_APPL_INFO_V95 table function - Retrieve appl_info logical data group snapshot information	1266
SNAPAPPL administrative view and SNAP_GET_APPL table function - Retrieve appl logical data group snapshot information .	1272
SNAP_GET_APPL_V95 table function - Retrieve appl logical data group snapshot information	1280
SNAPBP administrative view and SNAP_GET_BP table function - Retrieve bufferpool logical group snapshot information .	1287
SNAPBP_PART administrative view and SNAP_GET_BP_PART table function - Retrieve bufferpool_nodeinfo logical data group snapshot information	1293
SNAP_GET_BP_V95 table function - Retrieve bufferpool logical group snapshot information .	1297
SNAPCONTAINER administrative view and SNAP_GET_CONTAINER table function - Retrieve tablespace_container logical data group snapshot information	1300
SNAP_GET_CONTAINER_V91 table function - Retrieve tablespace_container logical data group snapshot information	1305
SNAPDB administrative view and SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group	1308
SNAPDB_MEMORY_POOL administrative view and SNAP_GET_DB_MEMORY_POOL table function - Retrieve database level memory usage information	1320
SNAP_GET_DB_V97 table function - Retrieve snapshot information from the dbase logical group	1324
SNAPDBM administrative view and SNAP_GET_DBM table function - Retrieve the dbm logical grouping snapshot information . .	1334
SNAPDBM_MEMORY_POOL administrative view and SNAP_GET_DBM_MEMORY_POOL table function - Retrieve database manager level memory usage information	1339
SNAP_GET_DBM_V95 table function - Retrieve the dbm logical grouping snapshot information	1342
SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG table function - Retrieve snapshot information from the detail_log logical data group	1345
SNAP_GET_DETAILLOG_V91 table function - Retrieve snapshot information from the detail_log logical data group	1349
SNAP_GET_DYN_SQL_V95 table function - Retrieve dynsql logical group snapshot information	1351

SNAPFCM administrative view and SNAP_GET_FCM table function - Retrieve the fcm logical data group snapshot information .	1355
SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function - Retrieve the fcm_node logical data group snapshot information	1358
SNAPHADR administrative view and SNAP_GET_HADR table function - Retrieve hadr logical data group snapshot information .	1361
SNAPLOCK administrative view and SNAP_GET_LOCK table function - Retrieve lock logical data group snapshot information .	1365
SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function - Retrieve lockwait logical data group snapshot information	1371
SNAPTbsp_RANGE administrative view and SNAP_GET_TBSP_RANGE table function - Retrieve range snapshot information	1377
SNAPSTMT administrative view and SNAP_GET_STMT table function - Retrieve statement snapshot information	1382
SNAP_GET_STO_PATHS	1388
SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS_V97 table function - Retrieve automatic storage path information	1390
SNAPTAB administrative view and SNAP_GET_TAB table function - Retrieve table logical data group snapshot information . . .	1393
SNAP_GET_TAB_V91	1397
SNAPTbsp administrative view and SNAP_GET_TBSP table function - Retrieve table space logical data group snapshot information	1399
SNAPTbsp_PART administrative view and SNAP_GET_TBSP_PART table function - Retrieve tablespace_nodeinfo logical data group snapshot information	1405
SNAP_GET_TBSP_PART_V97 table function - Retrieve tablespace_nodeinfo logical data group snapshot information	1411
SNAP_GET_TBSP_V91	1415
SNAPTbsp_QUIESCER administrative view and SNAP_GET_TBSP_QUIESCER table function - Retrieve quiescer table space snapshot information	1419
SNAPUTIL administrative view and SNAP_GET_UTIL table function - Retrieve utility_info logical data group snapshot information	1424
SNAP_WRITE_FILE procedure	1428
SNAPAGENT administrative view and SNAP_GET_AGENT table function - Retrieve agent logical data group application snapshot information	1430

SNAPAGENT_MEMORY_POOL administrative view and	
SNAP_GET_AGENT_MEMORY_POOL table function - Retrieve memory_pool logical data group snapshot information	1433
SNAPAPPL_INFO administrative view and	
SNAP_GET_APPL_INFO table function - Retrieve appl_info logical data group snapshot information	1438
SNAPAPPL administrative view and	
SNAP_GET_APPL table function - Retrieve appl logical data group snapshot information	1446
SNAPBP administrative view and	
SNAP_GET_BP table function - Retrieve bufferpool logical group snapshot information	1454
SNAPBP_PART administrative view and	
SNAP_GET_BP_PART table function - Retrieve bufferpool_nodeinfo logical data group snapshot information	1460
SNAPCONTAINER administrative view and	
SNAP_GET_CONTAINER table function - Retrieve tablespace_container logical data group snapshot information	1464
SNAPDB administrative view and	
SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group	1468
SNAPDB_MEMORY_POOL administrative view and	
SNAP_GET_DB_MEMORY_POOL table function - Retrieve database level memory usage information	1480
SNAPDBM administrative view and	
SNAP_GET_DBM table function - Retrieve the dbm logical grouping snapshot information	1484
SNAPDBM_MEMORY_POOL administrative view and	
SNAP_GET_DBM_MEMORY_POOL table function - Retrieve database manager level memory usage information	1489
SNAPDETAILLOG administrative view and	
SNAP_GET_DETAILLOG table function - Retrieve snapshot information from the detail_log logical data group	1492
SNAPFCM administrative view and	
SNAP_GET_FCM table function - Retrieve the fcm logical data group snapshot information	1496
SNAPFCM_PART administrative view and	
SNAP_GET_FCM_PART table function - Retrieve the fcm_node logical data group snapshot information	1499
SNAPHADR administrative view and	
SNAP_GET_HADR table function - Retrieve hadr logical data group snapshot information	1502
SNAPLOCK administrative view and	
SNAP_GET_LOCK table function - Retrieve lock logical data group snapshot information	1507
SNAPLOCKWAIT administrative view and	
SNAP_GET_LOCKWAIT table function - Retrieve lockwait logical data group snapshot information	1512

SNAPDB administrative view and	
SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group	1518
SNAPUTIL administrative view and	
SNAP_GET_UTIL table function - Retrieve utility_info logical data group snapshot information	1530
SNAPSTMT administrative view and	
SNAP_GET_STMT table function - Retrieve statement snapshot information	1534
SNAPSTORAGE_PATHS administrative view and	
SNAP_GET_STORAGE_PATHS_V97 table function - Retrieve automatic storage path information	1540
SNAPTAB administrative view and	
SNAP_GET_TAB table function - Retrieve table logical data group snapshot information	1544
SNAPTbsp administrative view and	
SNAP_GET_TBSP table function - Retrieve table space logical data group snapshot information	1548
SNAPTbsp_PART administrative view and	
SNAP_GET_TBSP_PART table function - Retrieve tablespace_nodeinfo logical data group snapshot information	1553
SNAPTbsp_QUIESCER administrative view and	
SNAP_GET_TBSP_QUIESCER table function - Retrieve quiescer table space snapshot information	1559
SNAPTbsp_RANGE administrative view and	
SNAP_GET_TBSP_RANGE table function - Retrieve range snapshot information	1563
SNAPUTIL administrative view and	
SNAP_GET_UTIL table function - Retrieve utility_info logical data group snapshot information	1568
SYSINSTALLROUTINES	1572
TBSP_UTILIZATION administrative view - Retrieve table space configuration and utilization information	1572
WLM_GET_ACTIVITY_DETAILS - Return detailed information about a specific activity	1575
WLM_GET_SERVICE_CLASS_AGENTS_V97 - List agents running in a service class.	1580
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97 - List of workload occurrences	1588
WLM_GET_SERVICE_SUBCLASS_STATS_V97 - return statistics of service subclasses	1591
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97 - Return a list of activities	1597
WLM_GET_WORKLOAD_STATS_V97 - return workload statistics	1601

Index	1605
------------------------	-------------

Figures

Tables

1. SQLCODE and information returned by the ADMIN_CMD procedure	6	32. Information returned by the ADMIN_GET_ENCRYPTION_INFO routine	203
2. Valid input keywords and parameter values	9	33. Information returned by ADMIN_GET_INDEX_COMPRESS_INFO	208
3. Valid file type modifiers for the export utility: All file formats	38	34. Information returned by ADMIN_GET_INDEX_INFO	211
4. Valid file type modifiers for the export utility: DEL (delimited ASCII) file format	39	35. Information returned for ADMIN_GET_MEM_USAGE	214
5. Valid file type modifiers for the export utility: IXF file format	41	36. Information returned by the ADMIN_GET_MSGS table function	216
6. Valid file type modifiers for the import utility: All file formats	64	37. Information returned by the ADMIN_GET_STORAGE_PATHS table function	218
7. Valid file type modifiers for the import utility: ASCII file formats (ASC/DEL)	67	38. Information returned for ADMIN_GET_TAB_COMPRESS_INFO	219
8. Valid file type modifiers for the import utility: ASC (non-delimited ASCII) file format	70	39. Information returned by ADMIN_GET_TAB_DICTIONARY_INFO	222
9. Valid file type modifiers for the import utility: DEL (delimited ASCII) file format	71	40. Information Returned by ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO	226
10. Valid file type modifiers for the import utility: IXF file format	72	41. Information returned for ADMINTEMPCOLUMNS administrative view and ADMIN_GET_TEMP_COLUMNS	235
11. IMPORT behavior when using codepage and usegraphiccodepage.	73	42. Information returned for ADMINTEMPTABLES administrative view and ADMIN_GET_TEMP_TABLES	239
12. LOAD REPLACE KEEPDICTIONARY	86	43. Information returned by the ADMIN_MOVE_TABLE stored procedure	250
13. LOAD REPLACE RESETDICTIONARY	87	44. Key and value pairs returned by the ADMIN_MOVE_TABLE stored procedure	250
14. LOAD TERMINATE dictionary management	107	45. SESSION.ADMIN_MOVE_TABLE returned by the operation REPORT	252
15. LOAD RESTART dictionary management	107	46. Information Returned by ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO	276
16. Valid file type modifiers for the load utility: All file formats	108	47. Information returned for ADMINTEMPCOLUMNS administrative view and ADMIN_GET_TEMP_COLUMNS	285
17. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL)	114	48. Information returned for ADMINTEMPTABLES administrative view and ADMIN_GET_TEMP_TABLES	289
18. Valid file type modifiers for the load utility: ASC file formats (Non-delimited ASCII).	118	49. Field names and values for the UNIX cron format	294
19. Valid file type modifiers for the load utility: DEL file formats (Delimited ASCII)	119	50. Example task schedules and the appropriate <i>schedule</i> argument values.	296
20. Valid file type modifiers for the load utility: IXF file format	120	51. Information returned by the ADMIN_TASK_LIST administrative view	296
21. LOAD behavior when using code page and usegraphiccodepage	122	52. Information returned by the ADMIN_TASK_STATUS administrative view	299
22. Supported access mode for CLASSIC table reorganization on nonpartitioned and partitioned table	140	53. Information returned by the AUDIT_ARCHIVE procedure and table function	304
23. Access modes supported and concurrent access allowed when the ON DATA PARTITION clause is specified with REORG INDEXES ALL	142	54. The information returned for AUDIT_LIST_LOGS	306
24. Supported access modes for index reorganization on partitioned and nonpartitioned table	142	55. Common SQL API shared input and output parameters	313
25. REORG KEEPDICTIONARY	147		
26. REORG KEEPDICTIONARY when LONGLOBDATA option is specified.. . . .	148		
27. REORG RESETDICTIONARY	149		
28. REORG RESETDICTIONARY when LONGLOBDATA option is specified.. . . .	149		
29. ADMIN_COPY_SCHEMA errortab format	197		
30. ADMIN_DROP_SCHEMA errortab format	199		
31. Row size limits	203		

56. Information returned by the DESIGN_ADVISOR - retrieve design advisor recommendations table function	326	82. Information returned by the MON_CURRENT_SQL administrative view	435
57. XML_INPUT options	327	83. Information returned by the MON_CURRENT_UOW administrative view	437
58. XML_OUTPUT description	328	84. Information returned by the MON_DB_SUMMARY administrative view	438
59. Information returned by the DBCFG administrative view and the DB_GET_CFG table function	365	85. Information returned by the MON_LOCKWAITS administrative view	442
60. Information returned by the DBMCFG administrative view and DBM_GET_CFG table function	370	86. Information returned by the MON_PKG_CACHE_SUMMARY administrative view	443
61. Information returned by the DB_MEMBERS table function	372	87. Information returned by the MON_SERVICE_SUBCLASS_SUMMARY administrative view	446
62. Information returned for DB2_CLUSTER_HOST_STATE administrative view and DB2_GET_CLUSTER_HOST_STATE	375	88. Information returned by the MON_TBSP_UTILIZATION administrative view	449
63. Information returned by the DB2_INSTANCE_ALERTS administrative view	376	89. Information returned by the MON_TRANSACTION_LOG_UTILIZATION administrative view	455
64. Information returned for DB2_MEMBER administrative view	379	90. Information returned by the MON_WORKLOAD_SUMMARY administrative view	456
65. Information returned for DB2_CF administrative view	380	91. Information returned for EVMON_FORMAT_UE_TO_XML.	474
66. Information returned for DB2_GET_INSTANCE_INFO table function	380	92. Information returned by EVMON_UPGRADE_TABLES to table SESSION.EVMON_UPGRADE_TABLES_RESULTSET	475
67. Information returned by the ENV_CF_SYS_RESOURCES administrative view	382	93. Information returned by the MON_FORMAT_LOCK_NAME table function	484
68. Possible values for the NAME and DATATYPE columns	382	94. Monitor elements that can be returned	484
69. ENV_FEATURE_INFO administrative view metadata	383	95. Information returned for MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW	486
70. Information returned by the ENV_GET_DB2_EDU_SYSTEM_RESOURCES routine.	385	96. Metric names returned by MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW for XML documents containing a system_metrics element type	487
71. Information returned by the ENV_GET_DB2_SYSTEM_RESOURCES table function	387	97. Metric names returned by MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW for XML documents containing an activity_metrics element type	488
72. Information returned by the ENV_GET_NETWORK_RESOURCES table function	389	98. Information returned for MON_FORMAT_XML_METRICS_BY_ROW	490
73. Information returned by the ENV_GET_REG_VARIABLES table function	390	99. Metric names returned by MON_FORMAT_XML_METRICS_BY_ROW for XML documents containing a system_metrics element type	491
74. Information returned by the ENV_GET_INSTANCE_CODE_LEVELS routine.	392	100. Metric names returned by MON_FORMAT_XML_METRICS_BY_ROW for XML documents containing an activity_metrics element type	498
75. Information returned for ENV_GET_SYSTEM_RESOURCES	396	101. Information returned for MON_FORMAT_XML_TIMES_BY_ROW	505
76. Information returned by the ENV_INST_INFO administrative view	399	102. Metric names returned by MON_FORMAT_XML_TIMES_BY_ROW for XML documents containing a system_metrics element type.	506
77. ENV_PROD_INFO administrative view metadata	400	103. Metric names returned by MON_FORMAT_XML_TIMES_BY_ROW for XML documents containing an activity_metrics element type	508
78. Information returned by the ENV_SYS_INFO administrative view	402		
79. Information returned by the EXPLAIN_GET_MSGS table function	404		
80. Information returned by the MON_BP_UTILIZATION administrative view	422		
81. Information returned by the MON_CONNECTION_SUMMARY administrative view	430		

104.	Information returned for MON_FORMAT_XML_WAIT_TIMES_BY_ROW	511	133.	Information returned for MON_GET_HADR	657
105.	Metric names returned by MON_FORMAT_XML_WAIT_TIMES_BY_ROW for XML documents containing a system_metrics element type	512	134.	Information returned for MON_GET_INDEX	663
106.	Metric names returned by MON_FORMAT_XML_WAIT_TIMES_BY_ROW for XML documents containing an activity_metrics element type	513	135.	Information returned for MON_GET_INDEX_USAGE_LIST	667
107.	Information returned by the MON_GET_ACTIVITY routine.	515	136.	Information returned by the MON_GET_INSTANCE routine	668
108.	Information returned for MON_GET_ACTIVITY_DETAILS	534	137.	Information returned by the MON_GET_LATCH table function	672
109.	Detailed metrics returned for MON_GET_ACTIVITY_DETAILS	534	138.	Information returned by the MON_GET_LOCKS table function	678
110.	Information returned by the MON_GET_AGENT routine	552	139.	Information returned for MON_GET_MEMORY_POOL	681
111.	Possible combinations for EVENT_STATE, EVENT_TYPE, EVENT_OBJECT, and EVENT_OBJECT_NAME column values	557	140.	Information returned for MON_GET_MEMORY_SET	684
112.	Information returned by the MON_GET_APPL_LOCKWAIT table function.	562	141.	Information returned for MON_GET_PAGE_ACCESS_INFO	686
113.	Information returned for MON_GET_AUTO_MAINT_QUEUE.	568	142.	Information returned for MON_GET_PKG_CACHE_STMT	688
114.	Information returned for MON_GET_AUTO_RUNSTATS_QUEUE	570	143.	Information returned for MON_GET_PKG_CACHE_STMT_DETAILS	701
115.	Information returned for MON_GET_BUFFERPOOL	572	144.	Detailed metrics returned for MON_GET_PKG_CACHE_STMT_DETAILS	701
116.	Information returned for MON_GET_CF	581	145.	Information returned by the MON_GET_QUEUE_STATS routine	713
117.	Information that is returned for the MON_GET_CF_CMD function.	583	146.	Information returned for MON_GET_REBALANCE_STATUS	716
118.	List of CF commands that are monitored by the MON_GET_CF_CMD function	583	147.	Information returned for MON_GET_ROUTINE	718
119.	Information that is returned for the MON_GET_CF_WAIT_TIME function	585	148.	Information returned for MON_GET_ROUTINE_DETAILS	734
120.	List of CF commands that are monitored by the MON_GET_CF_WAIT_TIME function	585	149.	Detailed metrics returned for MON_GET_ROUTINE_DETAILS	735
121.	Information returned for MON_GET_CONNECTION.	589	150.	Information returned for MON_GET_ROUTINE_EXEC_LIST	749
122.	Information returned for MON_GET_CONNECTION_DETAILS	604	151.	Information returned for MON_GET_RTS_RQST function	753
123.	Detailed metrics returned for MON_GET_CONNECTION_DETAILS	605	152.	Information returned by the MON_GET_SECTION.	755
124.	Information returned for MON_GET_CONTAINER	618	153.	Possible values for the OBJECT_TYPE parameter.	756
125.	Information returned by the MON_GET_DATABASE routine	620	154.	Information returned by the MON_GET_SECTION_OBJECT	757
126.	Information returned by the MON_GET_DATABASE_DETAILS routine	636	155.	Information returned for MON_GET_SECTION_ROUTINE.	759
127.	Detailed metrics returned for MON_GET_DATABASE_DETAILS	636	156.	Information returned for MON_GET_SERVERLIST	761
128.	Information returned for MON_GET_EXTENDED_LATCH_WAIT	651	157.	Information returned for MON_GET_SERVICE_SUBCLASS.	763
129.	Information returned for MON_GET_EXTENT_MOVEMENT_STATUS	652	158.	Information returned for MON_GET_SERVICE_SUBCLASS_DETAILS	780
130.	Information returned for MON_GET_FCM	653	159.	Detailed metrics returned for MON_GET_SERVICE_SUBCLASS_DETAILS	781
131.	Information returned for MON_GET_FCM_CONNECTION_LIST.	654	160.	Information returned by the MON_GET_SERVICE_SUBCLASS_STATS routine.	792
132.	Information returned for MON_GET_GROUP_BUFFERPOOL	656	161.	Information returned by the MON_GET_SERVICE_SUPERCLASS_STATS routine.	799
			162.	Information returned for MON_GET_TABLE	801
			163.	Information returned for MON_GET_TABLESPACE	806

164. Information returned by MON_GET_TABLESPACE QUIESCER	818	193. Information returned by the SNAPSUBSECTION administrative view and the SNAP_GET_SUBSECTION table function	949
165. Information returned by MON_GET_TABLESPACE_RANGE	821	194. Information returned by the SNAPSWITCHES administrative view and the SNAP_GET_SWITCHES table function	953
166. Information returned for MON_GET_TABLE_USAGE_LIST.	824	195. Information returned by the SNAPTAB_REORG administrative view and the SNAP_GET_TAB_REORG table function	958
167. Information returned for MON_GET_TRANSACTION_LOG	828	196. Information returned by the SNAPUTIL_PROGRESS administrative view and the SNAP_GET_UTIL_PROGRESS table function	963
168. Information returned for MON_GET_UNIT_OF_WORK	830	197. Information returned by the SNAPDYN_SQL administrative view and the SNAP_GET_DYN_SQL table function	968
169. Information returned for MON_GET_UNIT_OF_WORK_DETAILS	846	198. Information returned by the SNAPSUBSECTION administrative view and the SNAP_GET_SUBSECTION table function	973
170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS	847	199. Information returned by the SNAPSWITCHES administrative view and the SNAP_GET_SWITCHES table function	977
171. Information returned for MON_GET_USAGE_LIST_STATUS	863	200. Information returned by the SNAPTAB_REORG administrative view and the SNAP_GET_TAB_REORG table function	982
172. Information returned by the MON_GET_UTILITY routine	864	201. Information returned by the SNAPUTIL_PROGRESS administrative view and the SNAP_GET_UTIL_PROGRESS table function	987
173. Information returned for MON_GET_WORKLOAD	866	202. Information returned by the TOP_DYNAMIC_SQL administrative view.	988
174. Information returned for MON_GET_WORKLOAD_DETAILS	881	203. Information returned by the ANALYZE_LOG_SPACE procedure.	1002
175. Detailed metrics returned for MON_GET_WORKLOAD_DETAILS	881	204. Bitwise field identifiers	1004
176. Information returned by MON_GET_WORKLOAD_STATS.	892	205. Bitwise field identifiers	1006
177. Information returned for MON_GET_WORK_ACTION_SET_STATS	896	206. STMG_OBJECT_TYPE table	1009
178. Information returned for MON_SAMPLE_SERVICE_CLASS_METRICS	899	207. Specifications for option-value	1017
179. Information returned from MON_SAMPLE_WORKLOAD_METRICS	902	208. Status changes without invalid index:	1020
180. Information returned by the MQREADALL table function	908	209. Option-value pairs	1034
181. Information returned by the MQREADALLCLOB table function	911	210. Information returned for WLM_GET_CONN_ENV	1062
182. Information returned by the MQRECEIVEALL table function	916	211. Detailed metrics returned for WLM_GET_CONN_ENV	1062
183. Information returned by the MQRECEIVEALLCLOB table function	918	212. Information returned for WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES	1066
184. Information returned by the ADMIN_ROTATE_MASTER_KEY procedure	925	213. Information returned by the COMPILATION_ENV table function	1079
185. The information returned for AUTH_LIST_AUTHORITIES_FOR_AUTHID	928	214. Elements of a compilation environment returned by the COMPILATION_ENV table function	1079
186. Information returned by the AUTH_LIST_GROUPS_FOR_AUTHID table function	932	215. Information returned by the CONTACTGROUPS administrative view	1081
187. The result sets for AUTH_LIST_ROLES_FOR_AUTHID.	933	216. Information returned by the CONTACTS administrative view	1082
188. Information returned by the AUTHORIZATIONIDS administrative view	936	217. Information returned by the DB_HISTORY administrative view	1083
189. Information returned by the OBJECTOWNERS administrative view	937	218. OPERATION and OPERATIONTYPE values	1086
190. Information returned by the PRIVILEGES administrative view	938	219. Information returned by the DBPATHS administrative view and the ADMIN_LIST_DB_PATHS table function	1089
191. Information returned by the QUERY_PREP_COST administrative view	939	220. TYPE column values.	1090
192. Information returned by the SNAPDYN_SQL administrative view and the SNAP_GET_DYN_SQL table function	944		

221. Information returned by the NOTIFICATIONLIST administrative view	1094	249. Information returned by the GET_DB_CONFIG procedure	1189
222. Information returned by the PD_GET_DIAG_HIST table function	1096	250. Information returned by the GET_DBM_CONFIG table function	1190
223. Information returned by the PDLOGMSG_LAST24HOURS administrative view and the PD_GET_LOG_MSGS table function	1106	251. Information returned by the HEALTH_CONT_HI table function	1192
224. Information returned by the REORGCHK_IX_STATS procedure	1111	252. Information returned by the HEALTH_CONT_HI_HIS table function	1195
225. Information returned by the REORGCHK_TB_STATS procedure	1113	253. Information returned by the HEALTH_CONT_INFO table function	1197
226. Deprecated SQL administrative routines or views and their replacement routines or views	1120	254. Information returned by the HEALTH_DB_HI table function	1200
227. The result set for ADMIN_GET_DBP_MEM_USAGE	1125	255. Information returned by the HEALTH_DB_HI_HIS table function	1204
228. Information returned by ADMINTABCOMPRESSINFO administrative view and ADMIN_GET_TAB_COMPRESS_INFO.	1131	256. Information returned by the HEALTH_DB_HIC table function	1206
229. Information returned by ADMIN_GET_TAB_COMPRESS_INFO_V97	1137	257. Information returned by the HEALTH_DB_HIC_HIS table function	1208
230. Information returned by the ADMIN_GET_TAB_INFO_V95 table function	1140	258. Information returned by the HEALTH_DB_INFO table function	1210
231. Information returned by ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO_V97	1145	259. Information returned by the HEALTH_DBM_HI table function	1212
232. Information returned by the AM_BASE_RPT_RECOMS table function	1151	260. Information returned by the HEALTH_DBM_HI_HIS table function.	1214
233. Information returned by the AM_BASE_RPTS table function	1153	261. Information returned by the HEALTH_DBM_INFO table function	1216
234. Information returned by the AM_GET_LOCK_CHN_TB procedure	1155	262. Information returned by the HEALTH_GET_ALERT_ACTION_CFG table function	1219
235. General application information returned by the AM_GET_LOCK_RPT procedure	1157	263. Information returned by the HEALTH_GET_ALERT_CFG table function	1222
236. Locks held information returned by the AM_GET_LOCK_RPT procedure.	1159	264. Information returned by the HEALTH_GET_IND_DEFINITION table function	1225
237. Locks wait information returned by the AM_GET_LOCK_RPT procedure.	1161	265. Information returned by the HEALTH_TBS_HI table function.	1229
238. Information returned by the APPL_PERFORMANCE administrative view.	1166	266. Information returned by the HEALTH_TBS_HI_HIS table function	1234
239. Information returned by the APPLICATIONS administrative view	1168	267. Information returned by the HEALTH_TBS_INFO table function.	1235
240. Information returned by the BP_HITRATIO administrative view	1173	268. Information returned by the LOCKS_HELD administrative view	1237
241. Information returned by the BP_READ_IO administrative view	1175	269. Information returned by the LOCKWAITS administrative view	1240
242. Information returned by the BP_WRITE_IO administrative view	1177	270. Information returned by the LOG_UTILIZATION administrative view	1243
243. Information returned by the CONTAINER_UTILIZATION administrative view	1179	271. Information returned by the LONG_RUNNING_SQL administrative view	1245
244. Information returned by the DB_PARTITIONS table function	1182	272. Information returned by the REG_VARIABLES administrative view.	1247
245. ENV_SYS_RESOURCES administrative view metadata.	1183	273. Information returned by the REG_LIST_VARIABLES table function	1249
246. Possible values for the NAME column	1183	274. Information returned by the SNAPAGENT administrative view and the SNAP_GET_AGENT table function	1253
247. ENV_SYS_RESOURCES table function metadata.	1186	275. Information returned by the SNAPAGENT_MEMORY_POOL administrative view and the SNAP_GET_AGENT_MEMORY_POOL table function	1257
248. Possible values for the NAME column	1186		

276. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function .	1261	297. Information returned by the SNAPLOCK administrative view and the SNAP_GET_LOCK table function	1368
277. Information returned by the SNAP_GET_APPL_INFO_V95 table function.	1268	298. Information returned by the SNAPLOCKWAIT administrative view and the SNAP_GET_LOCKWAIT table function .	1374
278. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function.	1275	299. Information returned by the SNAPTbsp_RANGE administrative view and the SNAP_GET_TBSP_RANGE table function	1381
279. Information returned by the SNAP_GET_APPL_V95 table function	1282	300. Information returned by the SNAPSTMT administrative view and the SNAP_GET_STMT table function	1385
280. Information returned by the SNAPBP administrative view and the SNAP_GET_BP table function	1290	301. Information returned by the SNAP_GET_STO_PATHS table function	1389
281. Information returned by the SNAPBP_PART administrative view and the SNAP_GET_BP_PART table function	1296	302. Information returned by the SNAPSTORAGE_PATHS administrative view and the SNAP_GET_STORAGE_PATHS_V97 table function	1392
282. Information returned by the SNAP_GET_BP_V95 table function	1298	303. Information returned by the SNAPTAB administrative view and the SNAP_GET_TAB table function	1396
283. Information returned by the SNAPCONTAINER administrative view and the SNAP_GET_CONTAINER table function.	1304	304. Information returned by the SNAP_GET_TAB_V91 table function	1398
284. Information returned by the SNAP_GET_CONTAINER_V91 table function	1307	305. Information returned by the SNAPTbsp administrative view and the SNAP_GET_TBSP table function	1402
285. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function	1312	306. Information returned by the SNAPTbsp_PART administrative view and the SNAP_GET_TBSP_PART table function .	1408
286. Information returned by the SNAPDB_MEMORY_POOL administrative view and the SNAP_GET_DB_MEMORY_POOL table function	1323	307. Information returned by the SNAP_GET_TBSP_PART_V97 table function .	1412
287. Information returned by the SNAP_GET_DB_V97 table function	1327	308. Information returned by the SNAP_GET_TBSP_V91 table function	1416
288. Information returned by the SNAPDBM administrative view and the SNAP_GET_DBM table function	1337	309. Information returned by the SNAPTbsp_QUIESCER administrative view and the SNAP_GET_TBSP_QUIESCER table function	1423
289. Information returned by the SNAPDBM_MEMORY_POOL administrative view and the SNAP_GET_DBM_MEMORY_POOL table function	1341	310. Information returned by the SNAPUTIL administrative view and the SNAP_GET_UTIL table function	1427
290. Information returned by the SNAP_GET_DBM_V95 table function	1344	311. Information returned by the SNAPAGENT administrative view and the SNAP_GET_AGENT table function	1433
291. Information returned by the SNAPDETAILLOG administrative view and the SNAP_GET_DETAILLOG table function	1348	312. Information returned by the SNAPAGENT_MEMORY_POOL administrative view and the SNAP_GET_AGENT_MEMORY_POOL table function	1437
292. Information returned by the SNAP_GET_DETAILLOG_V91 table function	1350	313. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function .	1441
293. Information returned by the SNAP_GET_DYN_SQL_V95 table function .	1353	314. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function.	1449
294. Information returned by the SNAPFCM administrative view and the SNAP_GET_FCM table function	1357	315. Information returned by the SNAPBP administrative view and the SNAP_GET_BP table function	1458
295. Information returned by the SNAPFCM_PART administrative view and the SNAP_GET_FCM_PART table function .	1360		
296. Information returned by the SNAPHADR administrative view and the SNAP_GET_HADR table function	1364		

316. Information returned by the SNAPBP_PART administrative view and the SNAP_GET_BP_PART table function . . .	1463	331. Information returned by the SNAPSTORAGE_PATHS administrative view and the SNAP_GET_STORAGE_PATHS_V97 table function	1543
317. Information returned by the SNAPCONTAINER administrative view and the SNAP_GET_CONTAINER table function.	1468	332. Information returned by the SNAPTAB administrative view and the SNAP_GET_TAB table function	1547
318. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function	1473	333. Information returned by the SNAPTbsp administrative view and the SNAP_GET_TBSP table function	1550
319. Information returned by the SNAPDB_MEMORY_POOL administrative view and the SNAP_GET_DB_MEMORY_POOL table function	1483	334. Information returned by the SNAPTbsp_PART administrative view and the SNAP_GET_TBSP_PART table function .	1556
320. Information returned by the SNAPDBM administrative view and the SNAP_GET_DBM table function	1487	335. Information returned by the SNAPTbsp_QUIESCER administrative view and the SNAP_GET_TBSP_QUIESCER table function	1563
321. Information returned by the SNAPDBM_MEMORY_POOL administrative view and the SNAP_GET_DBM_MEMORY_POOL table function	1492	336. Information returned by the SNAPTbsp_RANGE administrative view and the SNAP_GET_TBSP_RANGE table function	1567
322. Information returned by the SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG table function . .	1496	337. Information returned by the SNAPUTIL administrative view and the SNAP_GET_UTIL table function	1571
323. Information returned by the SNAPFCM administrative view and the SNAP_GET_FCM table function	1499	338. Information returned by the TBSP_UTILIZATION administrative view.	1573
324. Information returned by the SNAPFCM_PART administrative view and the SNAP_GET_FCM_PART table function .	1502	339. Information returned for WLM_GET_ACTIVITY_DETAILS	1577
325. Information returned by the SNAPHADR administrative view and the SNAP_GET_HADR table function	1505	340. Elements returned	1577
326. Information returned by the SNAPLOCK administrative view and the SNAP_GET_LOCK table function	1510	341. Elements returned if applicable	1579
327. Information returned by the SNAPLOCKWAIT administrative view and the SNAP_GET_LOCKWAIT table function .	1515	342. Information returned by WLM_GET_SERVICE_CLASS_AGENTS_V97	1584
328. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function	1523	343. Possible combinations for EVENT_STATE, EVENT_TYPE, EVENT_OBJECT and EVENT_OBJECT_NAME column values . .	1588
329. Information returned by the SNAPUTIL administrative view and the SNAP_GET_UTIL table function	1533	344. Information returned for WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97	1594
330. Information returned by the SNAPSTMT administrative view and the SNAP_GET_STMT table function	1537	345. Information returned for WLM_GET_SERVICE_SUBCLASS_STATS_V97	1594
		346. Information returned by WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97	1599
		347. Information returned by WLM_GET_WORKLOAD_STATS_V97 . . .	1603

Built-in routines and views

Built-in administrative routines and views provide a simplified programmatic interface to administer and use databases and database objects through structured query language (SQL).

Built-in routines encompass procedures, scalar functions, and table functions.

You can run these built-in routines and views from an SQL-based application, a command line, or a command script.

Character and graphic strings that are used as parameters of routines or returned from the routines or views are always in OCTETS even when the environment is set to CODEUNITS32.

Best practices for calling built-in routines and views in applications

To help ensure your successful use of the built-in routines and views, certain coding practices are recommended. These practices are especially important because routines might change from release to release and also within releases, such as through fix packs, as enhancements are made.

When you issue a query to retrieve information by using a built-in routine or view, select specific columns instead of selecting all columns with a wildcard. For example, do not issue the following query:

```
SELECT * FROM TABLE(MON_GET_UNIT_OF_WORK(NULL,-1)) AS t
ORDER BY total_cpu_time DESC
```

Instead, name the result columns in the SELECT statement. This technique gives the application control over the number of result columns and the sequence in which they are returned. In the following rewrite of the previous query, the columns are named:

```
SELECT application_handle,
       uow_id,
       total_cpu_time,
       app_rqsts_completed_total,
       rqsts_completed_total
FROM TABLE(MON_GET_UNIT_OF_WORK(NULL,-1)) AS t
ORDER BY total_cpu_time DESC
```

Naming columns prevents problems if the sequence and number of columns in the routines change. The number of result columns that a routine returns might increase. If, for example, you provide only five host variables when the routine returns six result columns, your application will break.

In addition, the type and size of output parameters or result columns of routines might change. For example, a column might change from VARCHAR(8) to VARCHAR(128), or an INTEGER column might become a BIGINT column. If a variable that you use is too small, the data that you receive from the routine might be truncated.

To protect your C application from such changes, you can describe a prepared statement to determine which result columns are returned and what their types and sizes are. The following example shows how to describe a prepared statement:

```
strcpy(strStmt, "SELECT application_handle, uow_id, total_cpu_time
FROM TABLE(MON_GET_UNIT_OF_WORK(NULL,-1))
AS t ORDER BY total_cpu_time DESC");
EXEC SQL PREPARE stmt FROM :strStmt;
EXEC SQL DESCRIBE stmt INTO :pSqllda;
```

For an example of how to use the information that is returned in the SQL description area (SQLDA), see the RowDatamemoryAlloc function in the samples/c/tbread.sqc file.

For Java™ and .NET applications, you need to know the data type and size for a program, you can use metadata to determine which result columns are returned and what their types and sizes are, as shown in the following example:

```
ResultSet rs = pstmt.executeQuery();
ResultSetMetaData rsms = rs.getMetaData();
```

For an example of how to use the metadata of the result set, see the execPreparedQueryWithUnknownOutputColumn() method in the samples/java/jdbc/TbRead.java file.

Authorizations for using built-in routines and views

All built-in routines and views require specific privileges to run.

Built-in routines

For all built-in routines in the SYSPROC schema, you need EXECUTE privilege on the routine. You can use the following query to check whether your authorization ID, or a group or a role to which you belong, has EXECUTE privilege:

```
SELECT A.SPECIFICNAME, GRANTEE, GRANTEETYPE
FROM SYSCAT.ROUTINEAUTH A, SYSCAT.ROUTINES R
WHERE A.SCHEMA = R.ROUTINESCHEMA
AND A.SPECIFICNAME = R.SPECIFICNAME
AND A.SCHEMA = 'SYSPROC'
AND R.ROUTINENAME = 'routine_name'
AND A.EXECUTEAUTH <> 'N'
```

where *routine_name* is the name of the built-in routine.

If your authorization ID, or a group or a role to which you belong, is listed in the GRANTEE column, then you have access to the specified built-in routine.

Built-in views

For all built-in views in the SYSIBMADM schema, you need SELECT privilege on the view. You can use the following query to check whether your authorization ID, or a group or a role to which you belong, has SELECT privilege:

```
SELECT GRANTEE, GRANTEETYPE
FROM SYSCAT.TABAUTH
WHERE TABSCHEMA = 'SYSIBMADM'
AND TABNAME = 'view_name'
AND SELECTAUTH <> 'N'
```

where *view_name* is the name of the built-in view.

If your authorization ID, or a group or a role to which you belong, is listed in the GRANTEE column, then you have access to the specified built-in view.

Built-in views versus table functions

Built-in views provide a simplified application programming interface to database functions through SQL.

The built-in views fall into three categories:

- Views that are based on catalog views
- Views that are based on table functions with no input parameters
- Views that are based on table functions with one or more input parameters

A table function can return similar information as the built-in view, however you can use a table function to retrieve the information for a specific database on a specific database partition, a subset of all database partitions, or all database partitions.

The following examples illustrate the difference between using an built-in view, based on a table function with one or more input parameters, and using the corresponding table function:

- The **PDLOGMSGG_LAST24HOURS** view, which retrieves notification log messages, provides quick access to data from the previous 24 hours. By contrast, you can use the **PD_GET_LOG_MSGG** table function to retrieve data from a specified period of time.
- The snapshot monitor views, which are identified by names beginning with **SNAP**) provide access to data from each database partition. However, the snapshot monitor table functions, which are identified by names beginning with **SNAP_GET_**) provide the option to choose between data from a single database partition or a data subset from across all database partitions.
- The **ADMIN_TABINFO** view retrieves information for all tables in a database, which can significantly affect the performance of applications that use large databases. Instead, you can reduce the performance impact by using the **ADMIN_GET_TAB_INFO** table function and specifying the schema name, table name, or both, as input.

For built-in views based on table functions with one or more input parameters, both the built-in view and the table function can be used, each achieving a different goal:

The built-in views are always based on the most current version of the table functions. However, the column positions in the returned information may change from release to release to enable new information to be returned. Therefore, you should select specific columns from the built-in views or table functions, or describe your result set if your application uses a **SELECT *** statement.

Table functions with a version suffix (**_Vxx**) have been deprecated or discontinued. The deprecated functions might be discontinued in a future release. Therefore, you should change applications and scripts that use these table functions to invoke the corresponding table functions that have no version suffix.

Supported built-in SQL routines and views

You can use built-in routines and views to accomplish various administrative tasks.

For example, you can use built-in routines to reorganize a table, capture and retrieve monitor data, or retrieve the application ID of the current connection.

Built-in routines and views are grouped by function.

Administrative routines and ADMIN_CMD procedure

This grouping of routines and views can be used to administer your databases.

You can use dedicated procedures and functions, or the ADMIN_CMD procedure, which is used by applications to run administrative commands.

ADMIN_CMD - Run administrative commands

The ADMIN_CMD procedure is used by applications to run administrative commands using the SQL CALL statement.

Syntax

►► ADMIN_CMD (—*command-string*—) ◀◀

The schema is SYSPROC.

Procedure parameter

command-string

An input argument of type CLOB (2M) that specifies a single command that is to be executed.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

The procedure supports the following command line processor (CLP) commands:

- ADD CONTACT
- ADD CONTACTGROUP
- AUTOCONFIGURE
- BACKUP - online only
- DESCRIBE
- DROP CONTACT
- DROP CONTACTGROUP
- EXPORT
- FORCE APPLICATION
- IMPORT

- INITIALIZE TAPE
- LOAD
- PRUNE HISTORY/LOGFILE
- QUIESCE DATABASE
- QUIESCE TABLESPACES FOR TABLE
- REDISTRIBUTE
- REORG INDEXES/TABLE
- RESET ALERT CONFIGURATION
- RESET DATABASE CONFIGURATION
- RESET DATABASE MANAGER CONFIGURATION
- REWIND TAPE
- RUNSTATS
- SET TAPE POSITION
- UNQUIESCE DATABASE
- UPDATE ALERT CONFIGURATION
- UPDATE CONTACT
- UPDATE CONTACTGROUP
- UPDATE DATABASE CONFIGURATION
- UPDATE DATABASE MANAGER CONFIGURATION
- UPDATE HEALTH NOTIFICATION CONTACT LIST
- UPDATE HISTORY

Note: Some commands might have slightly different supported syntax when executed through the ADMIN_CMD procedure.

The procedure also supports the following commands which are not supported by the CLP:

- GET STMM TUNING
- UPDATE STMM TUNING

Usage notes

Retrieving command execution information:

- As the ADMIN_CMD procedure runs on the server, the utility messages are also created on the server. The **MESSAGES ON SERVER** option (refer to the specific command for further details) indicates that the message file is to be created on the server.
- Command execution status is returned in the SQLCA resulting from the CALL statement.
- If the execution of the administrative command is successful, and the command returns more than the execution status, the additional information is returned in the form of a result set (up to two result sets). For example, if the **EXPORT** command executes successfully, the returned result set contains information about the number of exported rows; however, if the **RUNSTATS** command executes successfully, no result set is returned. The result set information is documented with the corresponding command.
- If the execution of the administrative command is not successful, an SQL20397W warning message is returned by the ADMIN_CMD procedure along with a result set containing more details about the reason for the failure of the

administrative command. Any application that uses the ADMIN_CMD procedure should check the SQLCODE returned by the procedure. If the SQLCODE is ≥ 0 , the result set for the administrative command should be retrieved. The following table indicates what information might be returned depending on whether the **MESSAGES ON SERVER** option is used or not.

Table 1. SQLCODE and information returned by the ADMIN_CMD procedure

Administrative command execution status	MESSAGES ON SERVER option specified	MESSAGES ON SERVER option not specified
Successful	The SQLCODE returned is ≥ 0 : Additional information (result sets) returned, if any.	The SQLCODE returned is ≥ 0 : Additional information (result sets) returned, if any, but the MSG_RETRIEVAL and MSG_REMOVAL columns are NULL.
Failed	The SQLCODE returned 20397: Additional information (result sets) returned, but only the MSG_RETRIEVAL and MSG_REMOVAL columns are populated.	The SQLCODE returned is < 0 : No additional information (result sets) is returned.

- The result sets can be retrieved from the CLP or from applications such as JDBC and CLI applications, but not from embedded C applications.
- Case-sensitive names and double-byte character set (DBCS) names must be enclosed inside a backward slash and double quotation delimiter, for example, `\ " MyTabLe \ "`.

For all commands executed through the ADMIN_CMD, the user ID that established the connection to the database is used for authentication.

Any additional authority required, for example, for commands that need file system access on the database server, is documented in the reference information describing the command.

This procedure cannot be called from a user-defined function (SQLSTATE 38001) or a trigger.

ADD CONTACT command using the ADMIN_CMD procedure:

Adds a contact to the contact list which can be either defined locally on the system or in a global list. Contacts are users to whom processes such as the Scheduler and Health Monitor send messages.

The setting of the Database Administration Server (DAS) **contact_host** configuration parameter determines whether the list is local or global.

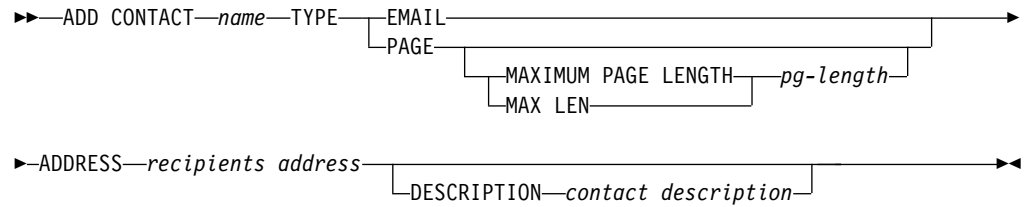
Authorization

None

Required connection

None. Local execution only: this command cannot be used with a remote connection.

Command syntax



Command parameters

ADD CONTACT *name*

The name of the contact that will be added. By default the contact will be added in the local system, unless the Db2® administration server configuration parameter **contact_host** points to another system.

TYPE Method of contact, which must be one of the following two:

EMAIL This contact wants to be notified by email at (**ADDRESS**).

PAGE This contact wants to be notified by a page sent to **ADDRESS**.

MAXIMUM PAGE LENGTH *pg-length*

If the paging service has a message-length restriction, it is specified here in characters.

The notification system uses the SMTP protocol to send the notification to the mail server specified by the Db2 Administration Server configuration parameter **smtp_server**. It is the responsibility of the SMTP server to send the email or call the pager.

ADDRESS *recipients-address*

The SMTP mailbox address of the recipient. For example, joe@somewhere.org. The **smtp_server** DAS configuration parameter must be set to the name of the SMTP server.

DESCRIPTION *contact description*

A textual description of the contact. This has a maximum length of 128 characters.

ADD CONTACTGROUP command using the **ADMIN_CMD** procedure:

Adds a new contact group to the list of groups defined on the local system. A contact group is a list of users and groups to whom monitoring processes such as the Scheduler and Health Monitor can send messages.

The setting of the Database Administration Server (DAS) **contact_host** configuration parameter determines whether the list is local or global.

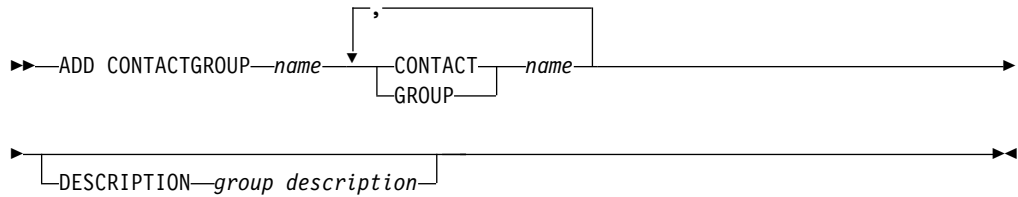
Authorization

None

Required connection

None. Local execution only: this command cannot be used with a remote connection.

Command Syntax



Command Parameters

ADD CONTACTGROUP *name*

Name of the new contact group, which must be unique among the set of groups on the system.

CONTACT *name*

Name of the contact which is a member of the group. A contact can be defined with the **ADD CONTACT** command after it has been added to a group.

GROUP *name*

Name of the contact group of which this group is a member.

DESCRIPTION *group description*

Optional. A textual description of the contact group.

AUTOCONFIGURE command using the **ADMIN_CMD** procedure:

Calculates and displays initial values for the buffer pool size, database configuration and database manager configuration parameters, with the option of applying these reported values.

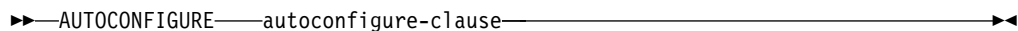
Authorization

SYSADM

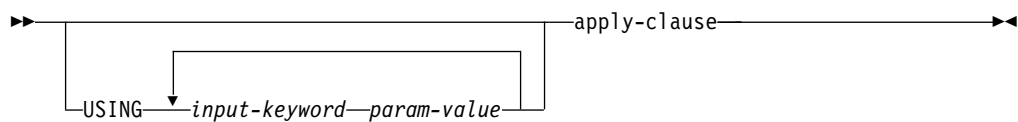
Required connection

Database

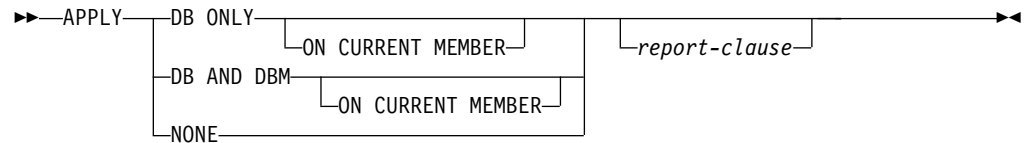
Command syntax



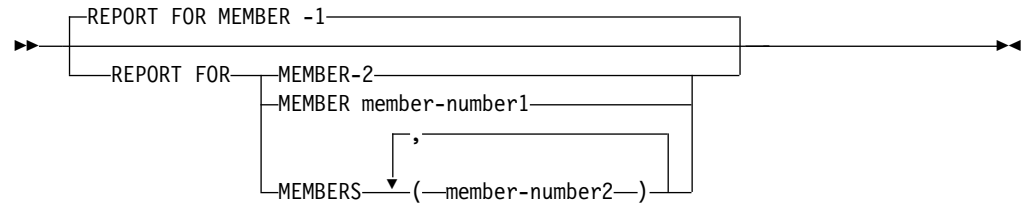
autoconfigure-clause:



apply-clause:



report-clause



Command parameters

USING *input-keyword param-value*

Table 2. Valid input keywords and parameter values

Keyword	Valid values	Default value	Explanation
mem_percent	1-100	<ul style="list-style-type: none"> When analytics is not enabled, the default value is 25 When analytics is enabled, mem_percent defaults to the higher of the following values: <ul style="list-style-type: none"> 25 80% divided by the number of database instances 	Percentage of instance memory that is assigned to the database. However, if the CREATE DATABASE command invokes the configuration advisor and you do not specify a value for mem_percent , the percentage is calculated based on memory usage in the instance and the system up to a maximum of 25% of the instance memory.
workload_type	simple, mixed, complex	mixed	Simple workloads tend to be I/O intensive and mostly transactions, whereas complex workloads tend to be CPU intensive and mostly queries.
num_stmts	1-1 000 000	10	Number of statements per unit of work
tpm	1-200 000	60	Transactions per minute
admin_priority	performance, recovery, both	both	Optimize for better performance (more transactions per minute) or better recovery time
is_populated	yes, no	yes	Is the database populated with data?
num_local_apps	0-5 000	0	Number of connected local applications
num_remote_apps	0-5 000	10	Number of connected remote applications

Table 2. Valid input keywords and parameter values (continued)

Keyword	Valid values	Default value	Explanation
isolation	RR, RS, CS, UR	RR	Maximum isolation level of applications connecting to this database (Repeatable Read, Read Stability, Cursor Stability, Uncommitted Read). It is only used to determine values of other configuration parameters. Nothing is set to restrict the applications to a particular isolation level and it is safe to use the default value.
bp_resizeable	yes, no	yes	Are buffer pools resizeable?
grp_tolerance_percent	1-100	5	Percentage of plus or minus tolerance to be used for the Summary Report when comparing configuration variable values to determine what group a member belongs.
analytics_env	automatic, yes, no	automatic	<p>Make recommendations for an analytic environment?</p> <p>Yes The Configuration Advisor will ignore the DB2_WORKLOAD registry variable and use this hint to determine recommendations for an analytics environment.</p> <p>No The Configuration Advisor will not make recommendations.</p> <p>Automatic The Db2 Configuration Advisor will use the value specified for the DB2_WORKLOAD registry variable to determine recommendations for an analytics environment.</p>

APPLY

DB ONLY

Displays the recommended values for the database configuration and the buffer pool settings based on the current database manager configuration. Applies the recommended changes to the database configuration and the buffer pool settings.

DB AND DBM

Displays and applies the recommended changes to the database manager configuration, the database configuration, and the buffer pool settings.

NONE Displays the recommended changes, but does not apply them.

ON CURRENT MEMBER

In a partitioned database environment or Db2 pureScale® environment, the Configuration Advisor updates the database configuration on all members by default. Specifying the **ON CURRENT MEMBER** option causes the Configuration Advisor to set the member-level configuration parameters on the current member determined by your connection, while the global-level

configuration parameters, that can be configured to be functional at only the global level, are set and affect all members.

The buffer pool changes are always applied to the system catalogs. Thus, all members are affected. The **ON CURRENT MEMBER** option is ignored for buffer pool recommendations.

REPORT FOR

Specifies the members to include in the report when used in a partitioned database environment or Db2 pureScale environment.

MEMBER -1

In a partitioned database environment or Db2 pureScale environment, the Configuration Advisor will report the member level configuration parameters computed changes recommended or made on the current member determined by your connection. This is the default if the REPORT FOR clause is not specified.

MEMBER -2

Indicates that the Configuration Advisor is to report the computed recommendations or changes for all members.

MEMBER member-number1

Specifies the number of the member the Configuration Advisor is to report the computed recommendations or changes for.

MEMBERS (member-number2...)

Specifies the numbers for each of the members the Configuration Advisor is to report the computed recommendations or changes for.

Usage notes

- This command makes configuration recommendations for the currently connected database and assumes that the database is the only active database on the instance. If you have not enabled the self tuning memory manager and you have more than one active database on the instance, specify a **mem_percent** value that reflects the database memory distribution. For example, if you have two active databases on the instance that should use 80% of the instance memory and should share the resources equally, specify 40% (80% divided by 2 databases) as the **mem_percent** value.
- If you have multiple instances on the same computer and the self tuning memory manager is not enabled, you should set a fixed value for **instance_memory** on each instance or specify a **mem_percent** value that reflects the database memory distribution. For example, if all active databases should use 80% of the computer memory and there are 4 instances each with one database, specify 20% (80% divided by 4 databases) as the **mem_percent** value.
- When explicitly invoking the Configuration Advisor with the **AUTOCONFIGURE** command, the setting of the **DB2_ENABLE_AUTOCONFIG_DEFAULT** registry variable will be ignored.
- Running the **AUTOCONFIGURE** command on a database will recommend enablement of the Self Tuning Memory Manager. However, if you run the **AUTOCONFIGURE** command on a database in an instance where **sheapthres** is not zero, sort memory tuning (**sortheap**) will not be enabled automatically. To enable sort memory tuning (**sortheap**), you must set **sheapthres** equal to zero using the

UPDATE DATABASE MANAGER CONFIGURATION command. Note that changing the value of **sheapthres** may affect the sort memory usage in your previously existing databases.

Compatibilities

For compatibility with previous versions:

- **NODE** and **DBPARTITIONNUM** can be specified in place of **MEMBER**, except when the **DB2_ENFORCE_MEMBER_SYNTAX** registry variable is set to ON.

BACKUP DATABASE command using the ADMIN_CMD procedure:

Creates a backup copy of a database or a table space. A backup of your Db2 database and related stored data must be created to prevent data loss if a database service outage occurs.

For information about the backup operations that are supported by Db2 database systems between different operating systems and hardware platforms, see “Backup and restore operations between different operating systems and hardware platforms”.

Scope

In a partitioned database environment, if no database partitions are specified, this command affects only the database partition on which it is run.

If the option to perform a partitioned backup is specified, the command can be called only on the catalog database partition. If the option specifies that all database partition servers are to be backed up, it affects all database partition servers that are listed in the `db2nodes.cfg` file. Otherwise, it affects the database partition servers that are specified on the command.

Authorization

One of the following authorities:

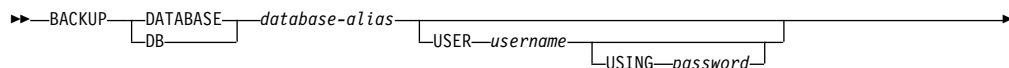
- SYSADM
- SYSCtrl
- SYSMAINT

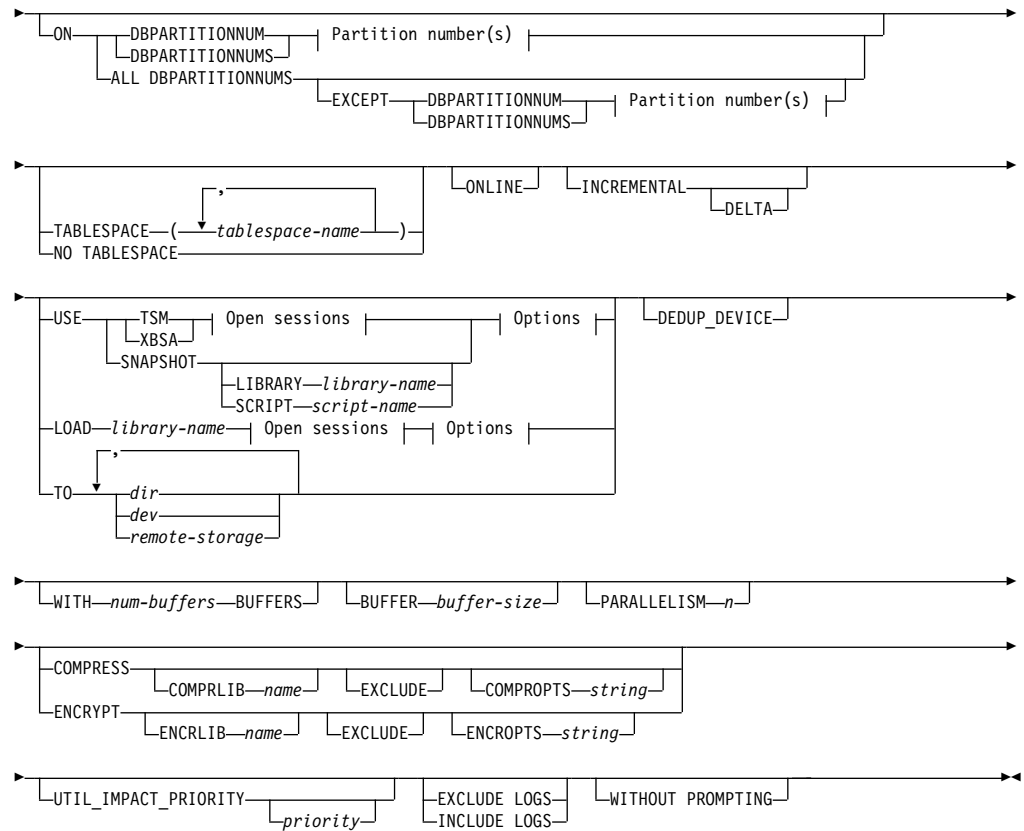
Required connection

Database. This command automatically establishes an exclusive connection to the specified database.

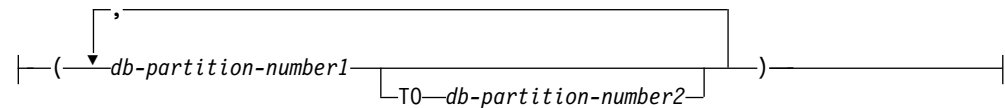
Note: If a connection to the specified database exists, that connection is terminated and a new connection established specifically for the backup operation. The connection is stopped at the completion of the backup operation.

Command syntax

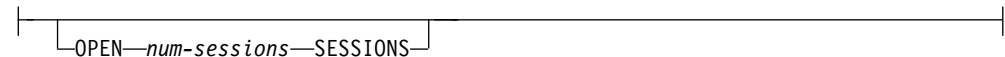




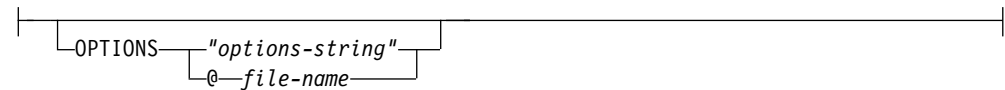
Partition number(s):



Open sessions:



Options:



Command parameters

DATABASE | **DB** *database-alias*

Specifies the alias of the database to back up.

USER *username*

Identifies the user name under which to back up the database.

USING *password*

The password that is used to authenticate the user name. If the password is omitted, the user is prompted to enter it.

ON Back up the database on a set of database partitions.

DBPARTITIONNUM *db-partition-number1*

Specifies a database partition number in the database partition list.

DBPARTITIONNUMS *db-partition-number1 TO db-partition-number2*

Specifies a range of database partition numbers so that all partitions from *db-partition-number1* up to and including *db-partition-number2* are included in the database partition list.

ALL DBPARTITIONNUMS

Specifies that the database is to be backed up on all partitions that are specified in the `db2nodes.cfg` file.

EXCEPT Specifies that the database is to be backed up on all partitions that are specified in the `db2nodes.cfg` file, except those specified in the database partition list.

DBPARTITIONNUM *db-partition-number1*

Specifies a database partition number in the database partition list.

DBPARTITIONNUMS *db-partition-number1 TO db-partition-number2*

Specifies a range of database partition numbers so that all partitions from *db-partition-number1* up to and including *db-partition-number2* are included in the database partition list.

TABLESPACE *tablespace-name*

A list of names that are used to specify the table spaces to be backed up.

NO TABLESPACE

Specifies a no tablespace backup. A no tablespace backup does not contain tablespace, but in particular, the history file that can be restored by using the **HISTORY FILE** options with the **RESTORE DATABASE** command.

ONLINE

Specifies online backup. The default is offline backup. Online backups are only available for databases that are configured with **logarchmeth1** enabled. During an online backup, Db2 obtains IN (Intent None) locks on all tables that exist in SMS table spaces as they are processed. S (share locks) is no longer held on LOB data in SMS table spaces during online backup.

INCREMENTAL

Specifies a cumulative (incremental) backup image. An incremental backup image is a copy of all database data that changed since the most recent successful, full backup operation.

DELTA Specifies a noncumulative (delta) backup image. A delta backup image is a copy of all database data that changed since the most recent successful backup operation of any type.

USE

TSM Specifies that the backup is to use Tivoli® Storage Manager (TSM) as the target device.

XBSA Specifies that the XBSA interface is to be used. Backup Services APIs (XBSA) are an open application programming interface for applications or facilities that need data storage management for backup or archiving purposes.

SNAPSHOT

Specifies that a snapshot backup is to be taken.

You cannot use the **SNAPSHOT** parameter with any of the following parameters:

- **TABLESPACE**
- **INCREMENTAL**
- **WITH *num-buffers* BUFFERS**
- **BUFFER**
- **PARALLELISM**
- **COMPRESS**
- **UTIL_IMPACT_PRIORITY**
- **SESSIONS**

The default behavior for a snapshot backup is a full database offline backup of all paths that make up the database. This backup includes all containers, local volume directory, database path (**DBPATH**), and primary log and mirror log paths (**INCLUDE LOGS** is the default for all snapshot backups unless **EXCLUDE LOGS** is explicitly stated).

LIBRARY *library-name*

Integrated into IBM® Db2 Server is a Db2 ACS API driver for the following storage hardware:

- IBM TotalStorage SAN Volume Controller
- IBM Enterprise Storage Server® Model 800
- IBM Storwize® V7000
- IBM System Storage® DS6000™
- IBM System Storage DS8000®
- IBM System Storage N Series
- IBM XIV®

If you have other storage hardware, and a Db2 ACS API driver for that storage hardware, you can use the **LIBRARY** parameter to specify the Db2 ACS API driver.

The value of the **LIBRARY** parameter is a fully qualified library file name.

SCRIPT *script-name*

The name of the executable script capable of performing a snapshot backup operation. The script name must be a fully qualified file name.

OPTIONS

"options-string"

Specifies options to be used for the backup operation. The string is passed exactly as it was entered, without the double quotation marks.

@ file-name

Specifies that the options to be used for the backup operation are contained in a file on the Db2 server. The string is passed to the vendor support library. The file must be a fully qualified file name.

You cannot use the **vendoropt** database configuration parameter to specify vendor-specific options for snapshot backup operations. You must use the **OPTIONS** parameter of the backup utilities instead.

OPEN num-sessions SESSIONS

The number of I/O sessions to create between the Db2 product and the TSM product or another backup vendor product. This parameter has no effect when you back up to tape, disk, or other local device. If you specify the **INCLUDE LOGS** parameter for an online backup, an extra session is created for the **OPEN num-sessions SESSIONS** parameter after the initial sessions are closed. If you are creating a Single System View (SSV) online backup, for each node that is backed up, an extra session is created for the **OPEN num-sessions SESSIONS** parameter after the initial sessions are closed. If you use this parameter with the TSM option, the number of entries that are created in the history file is equal to the number of sessions created.

TO dir | dev | remote-storage

A list of directory or tape device names. The full path on which the directory resides must be specified. If **USE TSM**, **TO**, and **LOAD** are omitted, the default target directory for the backup image is the current working directory of the client computer. This target directory or device must be locally addressable on the database server.

To back up to remote storage, such as IBM SoftLayer® Object Storage or Amazon Simple Storage Service (S3), specify a remote storage location by using a storage access alias. Local staging space is needed to temporarily store the backup image that is to be transferred to the remote storage server. Each backup session for the remote storage has a maximum size of 5 GB, which can produce a total database backup image size of 1 TB; refer to Remote storage requirements. The syntax of specifying a remote storage location is:

DB2REMOTE://<alias>//<storage-path>

In a partitioned database, the target directory or device must exist on all database partitions, and can optionally be a shared path. The directory or device name can be specified by using a database partition expression. For more information about database partition expressions, see “Automatic storage databases”.

This parameter can be repeated to specify the target directories and devices that the backup image spans. If more than one target is specified (target1, target2, and target3, for example), target1 is opened first. The media header and special files (including the configuration file, table space table, and history file) are placed in target1. All remaining targets are opened, and are then used in parallel during the backup operation. Because there is no general tape support on Windows operating systems, each type of tape device requires a unique device driver.

Use of tape devices or floppy disks might generate messages and prompts for user input. Valid response options are:

- c** Continue. Continue using the device that generated the warning message (for example, when a new tape has been mounted)

- d** Device terminate. Stop using only the device that generated the warning message (for example, when there are no more tapes)
- t** Terminate. Abort the backup operation.

If the tape system does not support the ability to uniquely reference a backup image, multiple backup copies of the same database must not be kept on the same tape.

LOAD *library-name*

The name of the shared library (DLL on Windows operating systems) containing the vendor backup and restore I/O functions to be used. It can contain the full path. If the full path is not given, it defaults to the path on which the user exit program resides.

DEDUP_DEVICE

Optimizes the format of the backup images for target storage devices that support data deduplication.

WITH *num-buffers* BUFFERS

The number of buffers to be used. If the number of buffers that you specify is not enough to create a successful backup, then the minimum value necessary to complete the backup is automatically chosen for this parameter. If you are backing up to multiple locations, you can specify a larger number of buffers to improve performance. If you specify the **COMPRESS** parameter, to help improve performance, you can add an extra buffer for each table space that you specify for the **PARALLELISM** parameter.

BUFFER *buffer-size*

The size, in 4 KB pages, of the buffer used when building the backup image. Db2 automatically chooses an optimal value for this parameter unless you explicitly enter a value. The minimum value for this parameter is eight pages.

If using tape with variable block size, reduce the buffer size to within the range that the tape device supports. Otherwise, the backup operation might succeed, but the resulting image might not be recoverable.

With most versions of Linux, using the default buffer size included with Db2 for backup operations to a SCSI tape device results in error SQL2025N, reason code 75. To prevent the overflow of Linux internal SCSI buffers, use this formula:

$$bufferpages \leq ST_MAX_BUFFERS * ST_BUFFER_BLOCKS / 4$$

where *bufferpages* is the value you want to use with the **BUFFER** parameter, and **ST_MAX_BUFFERS** and **ST_BUFFER_BLOCKS** are defined in the Linux kernel under the `drivers/scsi` directory.

PARALLELISM *n*

Determines the number of table spaces that can be read in parallel by the backup utility. Db2 automatically chooses an optimal value for this parameter unless you explicitly enter a value.

UTIL_IMPACT_PRIORITY *priority*

Specifies that the backup runs in throttled mode, with the priority specified. Throttling allows you to regulate the performance impact of the backup operation. Priority can be any number between 1 and 100, with 1 representing the lowest priority, and 100 representing the highest priority. If the **UTIL_IMPACT_PRIORITY** keyword is specified with no priority, the backup runs with the default priority of 50. If **UTIL_IMPACT_PRIORITY** is not

specified, the backup runs in unthrottled mode. An impact policy must be defined by setting the **util_impact_lim** configuration parameter for a backup to run in throttled mode.

COMPRESS | ENCRYPT

Indicates that the backup is to be compressed or encrypted. You cannot specify both parameters simultaneously. The **COMPRESS** and **ENCRYPT** parameters are synonyms and can be used interchangeably only when either **COMPRLIB** or **ENCRLIB** is specified too. If you specify **COMPRESS** without **COMPRLIB**, the default compression library `libdb2compr.so` is used for compression. If you specify **ENCRYPT** without **ENCRLIB**, the default encryption library `libdb2encr.so` is used for encryption. If you want to specify another library, you can use **COMPRESS** and **ENCRYPT** interchangeably. You can use either **COMPRLIB** or **ENCRLIB** to specify the `libdb2compr_encr.so` library. If the **encrlib** and **encropts** database configuration parameters are set to a non-NULL value, neither **COMPRLIB** or **ENCRLIB** nor **COMPROPTS** or **ENCROPTS** can be specified.

COMPRLIB | ENCRLIB *name*

Indicates the name of the library that is used during the compression or encryption process. For example, `db2compr.dll` for Windows; `libdb2compr.so` for Linux and UNIX operating systems. The name must be a fully qualified path that refers to a file on the server. If this parameter is not specified, the default Db2 compression library is used. If the specified library cannot be loaded, the backup operation fails.

EXCLUDE

Indicates that the library is not stored in the backup image.

COMPROPTS | ENCROPTS *string*

Describes a block of binary data that is passed to the initialization routine in the library. The database manager passes this string directly from the client to the server. Any issues of byte reversal or code page conversion are handled by the compression library. If the first character of the data block is '@', the rest of the data is interpreted as the name of a file on the server. The database manager then replaces the contents of the string with the contents of this file and passes this new value to the initialization routine. The maximum length for *string* is 1024 bytes.

For the default Db2 libraries `libdb2compr_encr.so` (compression and encryption) or `libdb2encr.so` (encryption only), the format of the **ENCROPTS** *string* is as follows:

Cipher=*cipher-name*:Mode=*mode-name*:Key Length=*key-length*:
Master Key Label=*label-name-1*...:Master Key Label=*label-name-n*

- Cipher is optional. Valid values are AES and 3DES (the default is AES).
- Mode is optional. The default is CBC.
- Key length is optional. Valid values for AES are 128, 192, and 256 (the default is 256), and the only valid value for 3DES is 168.
- Master key label is optional. The default is the database master key label.

If you are using other libraries, the format of the **ENCROPTS** *string* depends on those libraries.

EXCLUDE LOGS

Specifies that the backup image must not include any log files. When you perform an offline backup operation, logs are excluded whether this option is specified, except for snapshot backups. Logs are excluded by default in the following backup scenarios:

- Offline backup of a single-partitioned database.
- Online or offline backup of a multi-partitioned database, when not using a single system view backup.

If you specify the **EXCLUDE LOGS** with a snapshot backup, writes to log files are allowed during the backup. These log files are included by default in the snapshot backup, but are not usable for recovery. If this backup is restored, the log files must not be extracted from the backup. If the log path was set to the default when the backup was taken, then it is not possible to exclude the log files from being restored and they must be deleted manually after the backup is restored. If the log path was not the default, then the log files can be excluded at restore time by using the **LOGTARGET EXCLUDE** options with the **RESTORE DATABASE** command.

INCLUDE LOGS

Specifies that the backup image must include the range of log files that are required to restore and roll forward this image to some consistent point in time. This option is not valid for an offline backup, except for snapshot backups. **INCLUDE LOGS** always the default option for any online backup operation, except a multi-partitioned online backup where each database partition is backed up independently (that is, a non-single system view backup).

If any of the log files that are required for the backup have previously been backed up and are no longer in the log path, then the Db2 database manager retrieves them for backup from the overflow log path, if the path has been set. Otherwise, the database manager retrieves them for backup from the current log path or mirror log path. These log files are removed from the log path after the backup is complete.

WITHOUT PROMPTING

Specifies that the backup runs unattended, and that any actions that normally require user intervention returns an error message.

Note:

1. If the backup command indicates which partitions in a partitioned database are to be backed up, the backup operation is implicitly performed **WITHOUT PROMPTING**.
2. Options that are specified on the **BACKUP DATABASE** command in a partitioned database environment are applied on each partition individually. For example, if a backup operation is specified to **USE TSM OPEN 3 SESSIONS**, Db2 opens three TSM sessions on each partition.

Examples

1. In the following example, the database WSDB is defined on all four database partitions, numbered 0 - 3. The path /dev3/backup is accessible from all database partitions. Database partition 0 is the catalog partition. To perform an offline backup of all the WSDB database partitions to /dev3/backup, issue the following command from database partition 0:

```
db2 BACKUP DATABASE wsdb ON ALL DBPARTITIONNUMS TO /dev3/backup
```

The backup is performed simultaneously on all partitions. All four database partition backup images are stored in the /dev3/backup directory. This directory can be a shared directory that is accessible from more than one partition, or a locally mounted directory accessible from each partition individually, or a combination of both.

2. In the following example database SAMPLE is backed up to a TSM server that uses two concurrent TSM client sessions. Db2 calculates the optimal buffer size for this environment.

```
db2 backup database sample use tsm open 2 sessions with 4 buffers
```

3. In the following example, a table space-level backup of table spaces (syscatspace, userspace1) of database payroll is done to tapes.

```
db2 backup database payroll tablespace (syscatspace, userspace1) to
/dev/rmt0, /dev/rmt1 with 8 buffers without prompting
```

4. The **USE TSM OPTIONS** keywords can be used to specify the TSM information to use for the backup operation. The following example shows how to use the **USE TSM OPTIONS** keywords to specify a fully qualified file name:

```
db2 backup db sample use TSM options @/u/dmcinnis/myoptions.txt
```

The file myoptions.txt contains the following information: -fromnode=bar
-fromowner=dmcinnis

5. Following is a sample weekly incremental backup strategy for a recoverable database. It includes a weekly full database backup operation, a daily noncumulative (delta) backup operation, and a midweek cumulative (incremental) backup operation:

```
(Sun) db2 backup db sample use tsm
(Mon) db2 backup db sample online incremental delta use tsm
(Tue) db2 backup db sample online incremental delta use tsm
(Wed) db2 backup db sample online incremental use tsm
(Thu) db2 backup db sample online incremental delta use tsm
(Fri) db2 backup db sample online incremental delta use tsm
(Sat) db2 backup db sample online incremental use tsm
```

6. In the following example, three identical target directories are specified for a backup operation on database SAMPLE. This is suitable if the target file system is made up of multiple physical disks.

```
db2 backup database sample to /dev3/backup, /dev3/backup, /dev3/backup
```

The data is concurrently backed up to the three target directories, and three backup images are generated with extensions .001, .002, and .003.

7. In the following example, the database WSDb is defined on all four database partitions, numbered 0 - 3. Database partition 0 is the catalog partition. To perform an online backup of table space USERSPACE1 on database partitions 1 and 2, with the backup image to be stored on a TSM server, issue the following command from partition 0:

```
db2 BACKUP DATABASE wsdb ON DBPARTITIONNUMS (1, 2) TABLESPACE (USERSPACE1)
ONLINE USE TSM
```

8. Sample output that is generated to indicate the sqlcode returned by each partition.

Example 1

All partitions are successful (sqlcode >= 0)

```
$ db2 backup db foo on all dbpartitionnums tablespace(T1)
```

```
Part Result
```

```
----
```

```
0000 DB20000I The BACKUP DATABASE command completed successfully.
```

```
0001 SQL2430W The database backup succeeded, but the following
table spaces do not exist on this database partition: "T1".
```

Backup successful. The timestamp for this backup image is :
20040908010203

Example 2

One or more partitions fail (sqlcode < 0)

```
$ db2 backup db foo on all dbpartitionnums to /backups
```

Part Result

0000 SQL2001N The utility was interrupted. The output data may be incomplete.

0001 SQL2419N The target disk "/backups" has become full.

SQL2429N The database backup failed. The following database partitions returned errors: "1".

9. The following backups include the log directories in the image created:

```
db2 backup db sample use snapshot
```

```
db2 backup db sample online use snapshot
```

```
db2 backup db sample use snapshot INCLUDE LOGS
```

```
db2 backup db sample online use snapshot INCLUDE LOGS
```

10. The following backups will NOT include the log directories in the image created:

```
db2 backup db sample use snapshot EXCLUDE LOGS
```

```
db2 backup db sample online use snapshot EXCLUDE LOGS
```

11. The following command encrypts a backup image by using the default encryption options:

```
BACKUP DATABASE ccards ENCRYPT;
```

12. The following commands encrypt a backup image by using the default encryption library and explicitly provided encryption options:

```
BACKUP DATABASE ccards  
  ENCRYPT ENCRLIB 'libdb2encr.so'  
  ENCROPTS 'Cipher=3DES';
```

```
BACKUP DATABASE ccards  
  ENCRYPT ENCRLIB 'libdb2encr.so'  
  ENCROPTS 'Master Key Label=mylabel.mydb.myinstance.myserver';
```

13. The following command creates a backup image that is both compressed and encrypted:

```
BACKUP DATABASE ccards  
  ENCRYPT ENCRLIB libdb2compr_encr.so;
```

Usage notes

- The data in a backup cannot be protected by the database server. Make sure that backups are properly safeguarded, particularly if the backup contains LBAC-protected data.
- When backing up to tape, use of a variable block size is not supported. If you must use this option, ensure that you have tested procedures in place that enable you to recover successfully, using backup images that were created with a variable block size.
- The backup utility cannot be used with a Type 2 connection.
- When using a variable block size, you must specify a backup buffer size that is less than or equal to the maximum limit for the tape devices that you are using. For optimal performance, the buffer size must be equal to the maximum block size limit of the device being used.

- Snapshot backups should be complemented with regular disk backups in case of failure in the filer/storage system.
- As you regularly backup your database, you might accumulate large database backup images, many database logs, and load copy images, all of which might be taking up a large amount of disk space. Refer to “Managing recovery objects” for information about how to manage these recovery objects.
- You can use the **OPTIONS** parameter to enable backup operations in TSM environments that support proxy nodes. For more information, see the “Configuring a Tivoli Storage Manager client” topic.
- You can use the **DB2_BCKP_PAGE_VERIFICATION** registry variable to enable DMS and AS page validation during the backup.
- You can use the **DB2_BCKP_INCLUDE_LOGS_WARNING** registry variable to specify that some online backups can now succeed even if they have not successfully included all of the required logs.
- After you issue a **BACKUP DATABASE** command with the **ONLINE** option and the **INCLUDE LOGS** option, the resulting backed-up database image includes all the log files necessary to roll forward to the end of backup.

DESCRIBE command using the ADMIN_CMD procedure:

Displays metadata about the columns, indexes, and data partitions of tables or views. This command can also display metadata about the output of SELECT, CALL, or XQuery statements.

Use the **DESCRIBE** command to display information about any of the following items:

- Output of a SELECT, CALL, or XQuery statement
- Columns of a table or a view
- Indexes of a table or a view
- Data partitions of a table or view

Authorization

The authorization required depends on the type of information you want to display using the **DESCRIBE** command.

- If the SYSTOOLSTMPSPACE table space exists, one of the authorities shown in the following table is required.

Object to display information about	Privileges or authorities required
Output of a SELECT statement or XQuery statement	<p>At least one of the following authorizations:</p> <ul style="list-style-type: none"> • DATAACCESS authority • DBADM authority • SQLADM authority • EXPLAIN authority • For each object referenced in the statement: <ul style="list-style-type: none"> – SELECT privilege on each table, view or nickname – EXECUTE privilege on each routine – READ privilege on each global variable – USAGE privilege on each sequence – EXECUTE privilege on each module

Object to display information about	Privileges or authorities required
Output of a CALL statement	<p>Any of the following privileges or authorities:</p> <ul style="list-style-type: none"> • DATAACCESS authority • EXECUTE privilege on the stored procedure
Columns of a table or a view	<p>Any of the following privileges or authorities for the SYSCAT.COLUMNS system catalog table:</p> <ul style="list-style-type: none"> • SELECT privilege • ACCESSCTRL authority • DATAACCESS authority • DBADM authority • SECADM authority • SQLADM authority <p>If you want to use the SHOW DETAIL parameter, you also require any of these privileges or authorities on the SYSCAT.DATAPARTITIONEXPRESSION system catalog table.</p> <p>Because PUBLIC has all the privileges over declared temporary tables, you can use the command to display information about any declared temporary table that exists within your connection.</p>
Indexes of a table or a view	<p>Any of the following privileges or authorities on the SYSCAT.INDEXES system catalog table:</p> <ul style="list-style-type: none"> • SELECT privilege • ACCESSCTRL authority • DATAACCESS authority • DBADM authority • SECADM authority • SQLADM authority <p>If you want to use the SHOW DETAIL parameter, you also require EXECUTE privilege on the GET_INDEX_COLNAMES() UDF.</p> <p>Because PUBLIC has all the privileges over declared temporary tables, you can use the command to display information about any declared temporary table that exists within your connection.</p>

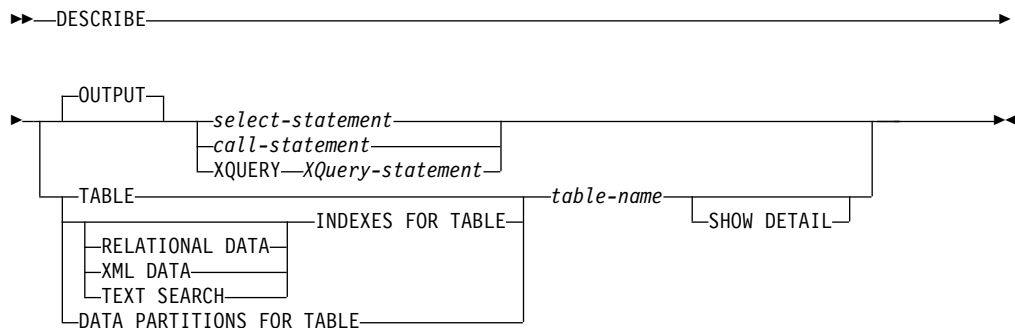
Object to display information about	Privileges or authorities required
Data partitions of a table or view	<p>Any of the following privileges or authorities on the SYSCAT.DATAPARTITIONS system catalog table:</p> <ul style="list-style-type: none"> • SELECT privilege • ACCESSCTRL authority • DATAACCESS authority • DBADM authority • SECADM authority • SQLADM authority <p>Because PUBLIC has all the privileges over declared temporary tables, you can use the command to display information about any declared temporary table that exists within your connection.</p>

- If the SYSTOOLSTMPSPACE table space does not exist, SYSADM or SYSCtrl authority is also required in addition to the one of the previously listed authorities.

Required connection

Database. If implicit connect is enabled, a connection to the default database is established.

Command syntax



Command parameters

OUTPUT Indicates that the output of the statement should be described. This keyword is optional.

select-statement | *call-statement* | **XQUERY** *XQuery-statement*

Identifies the statement about which information is wanted. The statement is automatically prepared by CLP. To identify an XQuery statement, precede the statement with the keyword **XQUERY**. A DESCRIBE OUTPUT statement only returns information about an implicitly hidden column if the column is explicitly specified as part of the SELECT list of the final result table of the query described.

TABLE *table-name*

Specifies the table or view to be described. The fully qualified name in the form *schema.table-name* must be used. An alias for the table cannot be used in place of the actual table. Information about implicitly hidden columns is returned, but SHOW DETAIL must be used to indicate which columns are implicitly hidden.

The **DESCRIBE TABLE** command lists the following information about each column:

- Column name
- Type schema
- Type name
- Length
- Scale
- Nulls (yes/no)

INDEXES FOR TABLE *table-name*

Specifies the table or view for which indexes need to be described. You can use the fully qualified name in the form *schema.table-name* or you can just specify the *table-name* and default schema will be used automatically. An alias for the table cannot be used in place of the actual table.

The **DESCRIBE INDEXES FOR TABLE** command lists the following information about each index of the table or view:

- Index schema
- Index name
- Unique rule
- Number of columns
- Index type

If the **DESCRIBE INDEXES FOR TABLE** command is specified with the **SHOW DETAIL** option, the index name is truncated when the index name is greater than 18 bytes. If no index type option is specified, information for all index types is listed: relational data index, index over XML data, and Text Search index. The output includes the following additional information:

- Index ID for a relational data index, an XML path index, an XML regions index, or an index over XML data
- Data Type for an index over XML data
- Hashed for an index over XML data
- Max VARCHAR Length for an index over XML data
- XML Pattern specified for an index over XML data
- Codepage for a text search index
- Language for a text search index
- Format specified for a text search index
- Update minimum for a text search index
- Update frequency for a text search index
- Collection directory for a text search index
- Column names of the indexes that are preceded by + for an ascending order, - for a descending order, and * for a random order.
- Expressions in square brackets following the generated column name for any part of an index key that is based on an expression.
- Whether the BUSINESS_TIME WITHOUT OVERLAPS clause is specified

Specify an index type to list information for only a specific index type. Specifying multiple index types is not supported.

Column name is truncated when it is greater than 8256 bytes.

RELATIONAL DATA

If the **RELATIONAL DATA** index type option is specified without the **SHOW DETAIL** option, only the following information is listed:

- Index schema
- Index name
- Unique rule
- Number of columns
- Null keys

If **SHOW DETAIL** is specified, the column names information is also listed.

XML DATA

If the **XML DATA** index type option is specified without the **SHOW DETAIL** option, only the following information is listed:

- Index schema
- Index name
- Unique rule
- Number of columns
- Index type

If **SHOW DETAIL** is specified, the following information for an index over XML data is also listed:

- Index ID
- Data type
- Hashed
- Max Varchar length
- XML Pattern
- Column names

TEXT SEARCH

If the **TEXT SEARCH** index type option is specified without the **SHOW DETAIL** option, only the following information is listed:

- Index schema
- Index name

If **SHOW DETAIL** is specified, the following text search index information is also listed:

- Column name
- Codepage
- Language
- Format
- Update minimum
- Update frequency
- Collection directory

If the **TEXT SEARCH** option is specified and a text search option is not installed or not properly configured, an error (SQLSTATE 42724) is returned.

See Db2 Text Search for information listed in the columns.

DATA PARTITIONS FOR TABLE *table-name*

Specifies the table or view for which data partitions need to be described. The information displayed for each data partition in the table includes; the partition identifier and the partitioning intervals. Results are ordered according to the partition identifier sequence. The fully qualified name in the form *schema.table-name* must be used. An alias for the table cannot be used in place of the actual table. The *schema* is the user name under which the table or view was created.

For the **DESCRIBE DATA PARTITIONS FOR TABLE** command, specifies that output include a second table with the following additional information:

- Data partition sequence identifier
- Data partition expression in SQL

SHOW DETAIL

For the **DESCRIBE TABLE** command, specifies that output include the following additional information:

- Whether a CHARACTER, VARCHAR or LONG VARCHAR column was defined as FOR BIT DATA
- Column number
- Distribution key sequence
- Code page
- Hidden attribute
- Default
- Table partitioning type (for tables partitioned by range this output appears after the original output)
- Partitioning key columns (for tables partitioned by range this output appears after the original output)
- Identifier of table space used for the index
- Periods that are defined on the table (for temporal tables this output appears after the original output)
- Whether versioning is enabled on the table (for temporal tables this output appears after the original output)

Examples

Describing the output of a SELECT Statement

The following example shows how to describe a SELECT statement:

```
db2 describe output select * from staff
```

Column Information

Number of columns: 7

Data Type	Length	Column Name	Name Length
500 SMALLINT	2	ID	2
449 VARCHAR	9	NAME	4
501 SMALLINT	2	DEPT	4
453 CHARACTER	5	JOB	3
501 SMALLINT	2	YEARS	5
485 DECIMAL	7,2	SALARY	6
485 DECIMAL	7,2	COMM	4

Describing the output of a CALL Statement

Given a stored procedure created with the statement:

```
CREATE PROCEDURE GIVE_BONUS (IN EMPNO INTEGER,
                             IN DEPTNO INTEGER,
                             OUT CHEQUE INTEGER,
                             INOUT BONUS DEC(6,0))
...
```

The following example shows how to describe the output of a CALL statement:

```
db2 describe output call give_bonus(123456, 987, ?, 15000.)
```

Column Information

Number of Columns: 2

Data Type	Length	Column Name	Name Length
497 INTEGER	4	CHEQUE	6
485 DECIMAL	6, 0	BONUS	5

If the procedure has one or more parameters of an array type, the output from the **DESCRIBE** command has one additional column, that indicates the maximum cardinality of array parameters. An empty value indicates that the parameter is not an array.

Given the array type and procedure created with the statements:

```
CREATE TYPE PRODUCT_LIST AS INTEGER ARRAY[100]
CREATE TYPE CUSTOMER_LIST AS INTEGER ARRAY[1000]
```

```
CREATE PROCEDURE DISCONTINUE_PROD (IN PROD_LIST PRODUCT_LIST,
                                    IN EFFECTIVE_DATE DATE,
                                    OUT NUM_PENDING_ORDERS INTEGER,
                                    OUT CUST_LIST CUSTOMER_LIST)
...
```

The following example shows how to describe the output of a CALL statement with array parameters. The only format difference with the previous example is the Max cardinality column.

```
db2 describe output call discontinue_prod(ARRAY[12, 34, 26], '04/13/2006', ?)
```

Column Information

Number of Columns: 2

SQL type	Type length	Column name	Name length	Max cardinality
497 INTEGER	4	NUM_PENDING_ORDERS	17	
497 INTEGER	10	CUSTOMER_LIST	13	1000

Describing the output of an XQuery Statement

Given a table named CUSTOMER that has a column named INFO of the XML data type, the following example shows how to describe an XQuery statement:

```
db2 describe xquery for $cust in db2-fn:xmlcolumn("CUSTOMER.INFO") return $cust
```

Column Information

Number of Columns: 1

SQL type	Type length	Column name	Name length
998 XML	0	1	1

If the keyword **XQUERY** is not specified, SQL0104N is returned.

```
db2 describe for $cust in db2-fn:xmlcolumn("CUSTOMER.INFO") return $cust
SQL0104N An unexpected token "for" was found following "DESCRIBE". Expected
tokens may include: "OUTPUT". SQLSTATE=42601
```

If the **DESCRIBE XQUERY** command is issued against a previous server that does not support the **XQUERY** option, the message DB21108E is returned to indicate that the functionality is not supported by the previous server.

Describing a Table

The following example shows how to describe a table:

```
db2 describe table user1.department
```

Table: USER1.DEPARTMENT

Column name	Data Type schema	Data Type name	Column length	Scale	Nulls
AREA	SYSIBM	SMALLINT	2		0 No
DEPT	SYSIBM	CHARACTER	3		0 No
DEPTNAME	SYSIBM	CHARACTER	20		0 Yes

The following example shows how to describe a table with details. If the table is partitioned, as in this example, additional details appear after the existing output. For a non-partitioned table, the additional table heading is not displayed:

```
db2 describe table user1.employee show detail
```

Column name	Data Type schema	Column number	Data Type name	Column length	Hidden
FIRST	SYSIBM	0	CHARACTER	10	No
LAST	SYSIBM	1	CHARACTER	10	No
EMPLOYEEENUM	SYSTEM	2	CHARACTER	10	Implicitly

Table is partitioned by range (ordered on the following column/s):

```
-----
LAST
FIRST
EMPLOYEEENUM
```

Describing a Table Index

The following example shows how to describe a table index. This command lists two relational data indexes, six xml data indexes, two text search indexes, and the system indexes:

```
db2 describe indexes for table user1.department
```

Index schema	Index name	Unique rule	Number of columns	Index type
SYSIBM	SQL070531145253450	D	-	XML DATA - REGIONS
SYSIBM	SQL070531145253620	U	1	XML DATA - PATH
USER1	RELIDX1	D	1	RELATIONAL DATA
USER1	RELIDX2	D	2	RELATIONAL DATA
SYSIBM	SQL070531145253650	P	1	RELATIONAL DATA
USER1	XMLIDX1	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154625650	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX2	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626000	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX3	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626090	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX4	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626190	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX5	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626290	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX6	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626400	D	1	XML DATA - VALUES PHYSICAL
USER1	TXTIDX1	-	1	TEXT SEARCH
USER1	TXTIDX2	-	1	TEXT SEARCH

The following command lists the relational data indexes for table USER1.DEPARTMENT:

```
db2 describe relational data indexes for table user1.department
```

Index schema	Index name	Unique rule	Number of columns	Null keys
SYSIBM	SQL070531145253650	P		1 Y
USER1	RELIDX1	D		1 N
USER1	RELIDX2	D		2 Y

The following command lists the indexes over XML data for table USER1.DEPARTMENT:

db2 describe xml data indexes for table user1.department

Index schema	Index name	Unique rule	Number of columns	Index type
SYSIBM	SQL070531145253450	D	-	XML DATA - REGIONS
SYSIBM	SQL070531145253620	U	1	XML DATA - PATH
USER1	XMLIDX1	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154625650	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX2	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626000	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX3	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626090	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX4	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626190	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX5	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626290	D	1	XML DATA - VALUES PHYSICAL
USER1	XMLIDX6	D	1	XML DATA - VALUES LOGICAL
SYSIBM	SQL070531154626400	D	1	XML DATA - VALUES PHYSICAL

The following command lists the text search index information for table USER1.DEPARTMENT:

db2 describe text search indexes for table user1.department

Index schema	Index name
USER1	TXIDX1
USER1	TXIDX2

The following command lists information about both partitioned and nonpartitioned indexes on the partitioned table myDPartT:

db2 describe indexes for table myDPartT

Index schema	Index name	Unique rule	Number of columns	Index Partitioning
NEWTON	IDXNDP	D	1	N
NEWTON	IDXDP	D	1	P

Describing Data Partitions

The following example shows how to describe data partitions:

db2 describe data partitions for table user1.sales

PartitionId	Inclusive (y/n) Low Value	Inclusive (y/n) High Value
0	Y 2001,1	Y 2001,3
1	N 2001,3	Y 2001,6
3	N 2001,6	Y 2001,9

Describing the data partitions with details returns the same output, as in the previous example, and includes an additional table showing the Partition ID and table space where the data for the data partition is stored, and the ID of the table space where the index is stored:

```
db2 describe data partitions for table user1.employee show detail
```

PartitionId	Inclusive (y/n) Low Value		Inclusive (y/n) High Value			
0	Y	MINVALUE,MINVALUE	Y	'beck','kevin'		
1	N	'beck','kevin'	N	'treece','jeff'		
2	Y	'treece','jeff'	Y	'zhang','liping'		
3	Y	'zyzyck',MINVALUE	Y	MAXVALUE,MAXVALUE		

PartitionId	PartitionName	TableSpId	LongTblSpId	IndexTblSpId	AccessMode	Status
0	PARTx	3	43	50	F	
1	PARTNew	13	13	13	N	A
2	PART3	31	33	35	F	
3	PART4	23	34	23	N	A

DROP CONTACT command using the ADMIN_CMD procedure:

Removes a contact from the list of contacts defined on the local system. A contact is a user to whom the Scheduler and Health Monitor send messages. The setting of the Database Administration Server (DAS) **contact_host** configuration parameter determines whether the list is local or global.

Authorization

None

Required connection

None. Local execution only: this command cannot be used with a remote connection.

Command syntax

```
➤➤—DROP CONTACT—name—————➤➤
```

Command parameters

CONTACT *name*

The name of the contact that will be dropped from the local system.

DROP CONTACTGROUP command using the ADMIN_CMD procedure:

Removes a contact group from the list of contacts defined on the local system. A contact group contains a list of users to whom the Scheduler and Health Monitor send messages. The setting of the Database Administration Server (DAS) **contact_host** configuration parameter determines whether the list is local or global.

Authorization

None

Required Connection

None

Command Syntax

►►—DROP CONTACTGROUP—*name*—————►

Command Parameters

CONTACTGROUP *name*

The name of the contact group that will be dropped from the local system.

EXPORT command using the ADMIN_CMD procedure:

Exports data from a database to one of several external file formats. The user specifies the data to be exported by supplying an SQL SELECT statement, or by providing hierarchical information for typed tables.

For more information, see “File type modifiers for the export utility” on page 38.

Authorization

One of the following authorities:

- DATAACCESS authority
- CONTROL or SELECT privilege on each participating table or view

Required connection

Database. If implicit connect is enabled, a connection to the default database is established. Utility access to Linux, UNIX, or Windows database servers from Linux, UNIX, or Windows clients must be a direct connection through the engine. Access must not be a direct connection through a Db2 Connect gateway or loop back environment.

Command syntax

EXPORT TO *filename* OF *filetype*

LOBS TO *lob-path*

LOBFILE *filename*

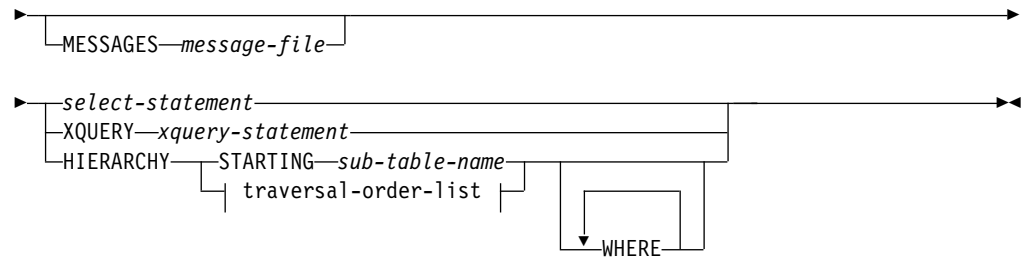
XML TO *xml-path*

XMLFILE *filename*

MODIFIED BY *filetype-mod*

XMLSAVESHEMA

METHOD N (*column-name*)



traversal-order-list:



Command parameters

TO *filename*

Specifies the name of the file to which data is to be exported. If the complete path to the file is not specified, the export utility uses the current directory and the default drive as the destination.

If the name of a file that exists is specified, the export utility overwrites the contents of the file; it does not append the information.

OF *filetype*

Specifies the format of the data in the output file:

- DEL (delimited ASCII format), which is used by various database manager and file manager programs.
- IXF (Integration Exchange Format, PC version) is a proprietary binary format.

LOBS TO *lob-path*

Specifies one or more paths to directories in which the LOB files are to be stored. There must be at least one file per LOB path, and each file must contain at least one LOB. The maximum number of paths that can be specified is 999. This setting implicitly activates the LOBSINFILE behavior.

LOBFILE *filename*

Specifies one or more base file names for the LOB files. When name space is exhausted for the first name, the second name is used, and so on. This setting implicitly activates the LOBSINFILE behavior.

When you create LOB files during an export operation, file names are constructed by appending the current base name from this list to the current path (from *lob-path*). Then, you must append a 3-digit sequence number to start, and the three character identifier *lob*. For example, if the current LOB path is the directory */u/foo/lob/path/*, and the current LOB file name is *bar*, the LOB files that are created are */u/foo/lob/path/bar.001.lob*, */u/foo/lob/path/bar.002.lob*, and so on. The 3-digit sequence number in the LOB file name grows to 4-digits once 999 is used, 4-digits grow to 5-digits once 9999 is used, and so on.

XML TO *xml-path*

Specifies one or more paths to directories in which the XML files are to be stored. There is at least one file per XML path, and each file contains at

least one XQuery Data Model (XDM) instance. If more than one path is specified, then XDM instances are distributed evenly among the paths.

XMLFILE *filename*

Specifies one or more base file names for the XML files. When name space is exhausted for the first name, the second name is used, and so on.

When you create XML files during an export operation, file names are constructed by appending the current base name from this list to the current path (from *xml-path*). Then, you must append a 3-digit sequence number, and the three character identifier xml. For example, if the current XML path is the directory /u/foo/xml/path/, and the current XML file name is bar, the XML files that are created are /u/foo/xml/path/bar.001.xml, /u/foo/xml/path/bar.002.xml, and so on.

MODIFIED BY *filetype-mod*

Specifies file type modifier options. See “File type modifiers for the export utility” on page 38.

XMLSAVESHEMA

Specifies that XML schema information must be saved for all XML columns. For each exported XML document that was validated against an XML schema when it was inserted, the fully qualified SQL identifier of that schema is stored as an (SCH) attribute inside the corresponding XML Data Specifier (XDS). If the exported document was not validated against an XML schema or the schema object no longer exists in the database, an SCH attribute will not be included in the corresponding XDS.

The schema and name portions of the SQL identifier are stored as the "OBJECTSCHEMA" and "OBJECTNAME" values in the row of the SYSCAT.XSROBJECTS catalog table corresponding to the XML schema.

The **XMLSAVESHEMA** option is not compatible with XQuery sequences that do not produce well-formed XML documents.

METHOD N *column-name*

Specifies one or more column names to be used in the output file. If this parameter is not specified, the column names in the table are used. This parameter is valid only for IXF files, but is not valid when you export hierarchical data.

MESSAGES *message-file*

Specifies the destination for warning and error messages that occur during an export operation. If the file exists, the export utility appends the information. If *message-file* is omitted, the messages are written to standard output.

select-statement

Specifies the SELECT or XQUERY statement that returns the data to be exported. If the statement causes an error, a message is written to the message file (or to standard output). If the error code is one of SQL0012W, SQL0347W, SQL0360W, SQL0437W, or SQL1824W, the export operation continues; otherwise, it stops.

If the SELECT statement is in the form of SELECT * FROM *tablename* and the table contains implicitly hidden columns, you must explicitly specify whether data for the hidden columns is included in the export operation. Use one of the following methods to indicate whether data for hidden columns is included:

- Use one of the hidden column file type modifiers: specify **implicitlyhiddeninclude** when the export contains data for the hidden columns, or **implicitlyhiddenmissing** when the export does not.
db2 export to t.del of del modified by implicitlyhiddeninclude
select * from t
- Use the DB2_DMU_DEFAULT registry variable on the client-side to set the default behavior when data movement utilities encounter tables with implicitly hidden columns.
db2set DB2_DMU_DEFAULT=IMPLICITLYHIDDENINCLUDE
db2 export to t.del of del select * from t

HIERARCHY STARTING *sub-table-name*

Using the default traverse order (OUTER order for ASC or DEL files, or the order that is stored in PC/IXF data files), export a sub-hierarchy starting from *sub-table-name*.

HIERARCHY traversal-order-list

Export a sub-hierarchy by using the specified traverse order. All sub-tables must be listed in PRE-ORDER fashion. The first sub-table name is used as the target table name for the SELECT statement.

Examples

The following example shows how to export information from the STAFF table in the SAMPLE database to the file myfile.ixf. The output is in IXF format. You must be connected to the SAMPLE database before you issue the command. The index definitions (if any) are stored in the output file except when the database connection is made through Db2 Connect.

```
db2 export to myfile.ixf of ixf messages msgs.txt select * from staff
```

The following example shows how to export the information about employees in Department 20 from the STAFF table in the SAMPLE database. The output is in IXF format and goes into the awards.ixf file. You must first connect to the SAMPLE database before you issue the command. Also, the actual column name in the table is 'dept' instead of 'department'.

```
db2 export to awards.ixf of ixf messages msgs.txt select * from staff
where dept = 20
```

The following example shows how to export LOBs to a DEL file:

```
db2 export to myfile.del of del lobs to mylobs/
lobfile lobs1, lobs2 modified by lobsinfile
select * from emp_photo
```

The following example shows how to export LOBs to a DEL file, specifying a second directory for files that might not fit into the first directory:

```
db2 export to myfile.del of del
lobs to /db2exp1/, /db2exp2/ modified by lobsinfile
select * from emp_photo
```

The following example shows how to export data to a DEL file. This step is done by using a single quotation mark as the string delimiter, a semicolon as the column delimiter, and a comma as the decimal point. The same convention must be used when you import data back into the database:

```
db2 export to myfile.del of del
modified by chardel'' coldel; decpt,
select * from staff
```

Usage notes

- Be sure to complete all table operations and release all locks before you start an export operation. This step can be done by issuing a COMMIT after you close all cursors opened WITH HOLD, or by issuing a ROLLBACK.
- Table aliases can be used in the SELECT statement.
- You might encounter the SQL27981W message when it does not seem applicable, such as when the EXPORT table is not partitioned. You can safely ignore this warning in this case. This warning message might be returned if the SELECT statement of the EXPORT command includes the word ' from ' before the SQL keyword FROM.
- The messages that are placed in the message file include the information that is returned from the message retrieval service. Each message begins on a new line.
- PC/IXF import must be used to move data between databases. If character data that contains row separators is exported to a delimited ASCII (DEL) file and processed by a text transfer program, fields that contain the row separators shrinks or expands.
- The file copying step is not necessary if the source and the target databases are both accessible from the same client.
- Db2 Connect can be used to export tables from DRDA servers such as Db2 for z/OS®, Db2 for VM and VSE, and Db2 for OS/400®. Only PC/IXF export is supported.
- When you export to the IXF format, if identifiers exceed the maximum size that is supported by the IXF format, the export succeeds. However, the resulting datafile cannot be used by a subsequent import operation by using the CREATE mode (SQL27984W).
- When you export to the IXF format, the export utility does not maintain column-organized table metadata that is needed to re-create the column-organized table during a subsequent import operation by using the CREATE mode.
- When you export to a diskette on Windows, and the table that has more data than the capacity of a single diskette, the system prompts for another diskette. Multiple-part PC/IXF files (also known as multi-volume PC/IXF files, or logically split PC/IXF files), are generated and stored in separate diskettes. In each file, with the exception of the last, there is a Db2 CONTINUATION RECORD (or "AC" Record in short) written. This record indicates that the files are logically split and shows where to look for the next file. The files can then be transferred to an AIX® system to be read by the import and load utilities. The export utility will not create multiple-part PC/IXF files when invoked from an AIX system. For detailed usage, see the **IMPORT** command or **LOAD** command.
- The export utility stores the NOT NULL WITH DEFAULT attribute of the table in an IXF file if the SELECT statement provided is in the form SELECT * FROM tablename.
- When you export typed tables, subselect statements can be expressed only by specifying the target table name and the **WHERE** clause. Fullselect and *select-statement* cannot be specified when you export a hierarchy.
- For file formats other than IXF, it is recommended that you specify the traversal order list. The list tells Db2 how to traverse the hierarchy, and what sub-tables to export. If this list is not specified, all tables in the hierarchy are exported, and the default order is the OUTER order. The alternative is to use the default order, which is the order that is given by the OUTER function.
- Use the same traverse order during an import operation. The load utility does not support loading hierarchies or sub-hierarchies.

- When you export data from a table that has protected rows, the LBAC credentials that are held by the session authorization id might limit the rows that are exported. If the session authorization ID does not have read access to the rows, then they will not be exported. No error or warning is given.
- If the LBAC credentials that are held by the session authorization ID do not allow reading from one or more protected columns included in the export, then the export fails and an error (SQLSTATE 42512) is returned.
- When you run Data Movement utilities such as **export** and **db2move**, the query compiler might determine that the underlying query runs more efficiently against an MQT than the base table or tables. In this case, the query executes against a refresh deferred MQT, and the result of the utilities might not accurately represent the data in the underlying table.
- Export packages are bound by using the DATETIME ISO format. Thus, all date/time/timestamp values are converted into ISO format when cast to a string representation. Since the CLP packages are bound by using the DATETIME LOC format (locale-specific format), you might see inconsistent behavior between CLP and export if the CLP DATETIME format is different from ISO. For instance, the following SELECT statement might return expected results:

```
db2 select col2 from tab1 where char(col2)='05/10/2005';
COL2
-----
05/10/2005
05/10/2005
05/10/2005
3 record(s) selected.
```

But an export command that uses the same select clause will not:

```
db2 export to test.del of del select col2 from test
where char(col2)='05/10/2005';
Number of rows exported: 0
```

Now, replacing the LOCALE date format with ISO format gives the expected results:

```
db2 export to test.del of del select col2 from test
where char(col2)='2005-05-10';
Number of rows exported: 3
```

File type modifiers for the export utility

Table 3. Valid file type modifiers for the export utility: All file formats

Modifier	Description
lobsinfile	<p><i>lob-path</i> specifies the path to the files that contain LOB data.</p> <p>Each path contains at least one file that contains at least one LOB pointed to by a Lob Location Specifier (LLS) in the data file. The LLS is a string representation of the location of a LOB in a file that is stored in the LOB file path. The format of an LLS is <i>filename.ext.nnn.mmm/</i>, where <i>filename.ext</i> is the name of the file that contains the LOB, <i>nnn</i> is the offset in bytes of the LOB within the file, and <i>mmm</i> is the length of the LOB in bytes. For example, if the string db2exp.001.123.456/ is stored in the data file, the LOB is at offset 123 in the file db2exp.001, and is 456 bytes long.</p> <p>If you specify the lobsinfile modifier when you use EXPORT, the LOB data is placed in the locations that are specified by the LOBS TO clause. Otherwise the LOB data is sent to the data file directory. The LOBS TO clause specifies one or more paths to directories in which the LOB files are to be stored. There is at least one file per LOB path, and each file contains at least one LOB. The LOBS TO or LOBFILE options implicitly activates the LOBSINFILE behavior.</p> <p>To indicate a null LOB, enter the size as -1. If the size is specified as 0, it is treated as a 0 length LOB. For null LOBS with length of -1, the offset and the file name are ignored. For example, the LLS of a null LOB might be db2exp.001.7.-1/.</p>
implicitlyhiddeninclude	<p>This modifier is used with SELECT * queries and specifies that the data in implicitly hidden columns is exported even though that data is not included in the result of the SELECT * query. This modifier cannot be used with the implicitlyhiddenmissing modifier.</p> <p>If this modifier is used and the query is not a SELECT *, then an error is returned (SQLCODE SQL3526N).</p> <p>This modifier does not apply to the hidden RANDOM_DISTRIBUTION_KEY column of a random distribution tables that uses the random by generation method. The column must be explicitly referenced in the query to be included in the exported data.</p>
implicitlyhiddenmissing	<p>This modifier is used with SELECT * queries and specifies that the data in implicitly hidden columns is not exported. This modifier cannot be used with the implicitlyhiddeninclude modifier.</p> <p>If this modifier is used and the query is not a SELECT *, then an error is returned (SQLCODE SQL3526N).</p> <p>This modifier does not apply to the hidden RANDOM_DISTRIBUTION_KEY column of a random distribution tables that uses the random by generation method. The column must be explicitly referenced in the query to be included in the exported data.</p>
xmlinsefiles	Each XQuery Data Model (XDM) instance is written to a separate file. By default, multiple values are concatenated together in the same file.
lobsinsefiles	Each LOB value is written to a separate file. By default, multiple values are concatenated together in the same file.
xmlnodeclaration	XDM instances are written without an XML declaration tag. By default, XDM instances are exported with an XML declaration tag at the beginning that includes an encoding attribute.

Table 3. Valid file type modifiers for the export utility: All file formats (continued)

Modifier	Description
xmlchar	XDM instances are written in the character code page. The character codepage is the value that is specified by the codepage file type modifier, or the application code page if it is not specified. By default, XDM instances are written out in Unicode.
xmlgraphic	If the xmlgraphic modifier is specified with the EXPORT command, the exported XML document is encoded in the UTF-16 code page regardless of the application code page or the codepage file type modifier.

Table 4. Valid file type modifiers for the export utility: DEL (delimited ASCII) file format

Modifier	Description
chardelx	<p><i>x</i> is a single character string delimiter. The default value is a double quotation mark ("). The specified character is used in place of double quotation marks to enclose a character string.² If you want to explicitly specify the double quotation mark as the character string delimiter, it must be specified as follows:</p> <pre>modified by charde1""</pre> <p>The single quotation mark (') can also be specified as a character string delimiter as follows:</p> <pre>modified by charde1''</pre>
codepage= <i>x</i>	<p><i>x</i> is an ASCII character string. The value is interpreted as the code page of the data in the output data set. Converts character data from this code page from the application code page during the export operation.</p> <p>For pure DBCS (graphic), mixed DBCS, and EUC, delimiters are restricted to the range of x00 to x3F, inclusive.</p>
coldelx	<p><i>x</i> is a single character column delimiter. The default value is a comma (,). The specified character is used in place of a comma to signal the end of a column.²</p> <p>In the following example, colde1; causes the export utility to use the semicolon character (;) as a column delimiter for the exported data:</p> <pre>db2 "export to temp of del modified by colde1; select * from staff where dept = 20"</pre>
decplusblank	Plus sign character. Causes positive decimal values to be prefixed with a blank space instead of a plus sign (+). The default action is to prefix positive decimal values with a plus sign.
decptx	<i>x</i> is a single character substitute for the period as a decimal point character. The default value is a period (.). The specified character is used in place of a period as a decimal point character. ²
nochardel	<p>Column data is not surrounded by character delimiters. This option must not be specified if the data is intended to be imported or loaded by using Db2. It is provided to support vendor data files that do not have character delimiters. Improper usage might result in data loss or corruption.</p> <p>This option cannot be specified with charde1x or nodoubledel. These options are mutually exclusive.</p>
nodoubledel	Suppresses recognition of double character delimiters. ²

Table 4. Valid file type modifiers for the export utility: DEL (delimited ASCII) file format (continued)

[illegible]

Table 4. Valid file type modifiers for the export utility: DEL (delimited ASCII) file format (continued)

Modifier	Description
timestampformat="x"	<p>x is the format of the time stamp in the source file.⁴ Valid time stamp elements are:</p> <p>YYYY Year (4 digits in the range 0000 - 9999)</p> <p>M Month (1 or 2 digits in the range 1 - 12)</p> <p>MM Month (2 digits in the range 01 - 12; mutually exclusive with M and MMM)</p> <p>MMM Month (three-letter case-insensitive abbreviation for the month name; mutually exclusive with M and MM)</p> <p>D Day (1 or 2 digits in the range 1 - 31)</p> <p>DD Day (2 digits in the range 01 - 31; mutually exclusive with D)</p> <p>DDD Day of the year (3 digits in the range 001 - 366; mutually exclusive with other day or month elements)</p> <p>H Hour (1 or 2 digits in the range 0 - 12 for a 12-hour system, and 0 - 24 for a 24-hour system)</p> <p>HH Hour (2 digits in the range 00 - 12 for a 12-hour system, and 00 - 24 for a 24-hour system; mutually exclusive with H)</p> <p>M Minute (1 or 2 digits in the range 0 - 59)</p> <p>MM Minute (2 digits in the range 00 - 59; mutually exclusive with M, minute)</p> <p>S Second (1 or 2 digits in the range 0 - 59)</p> <p>SS Second (2 digits in the range 00 - 59; mutually exclusive with S)</p> <p>SSSSS Second of the day after midnight (5 digits in the range 00000 - 86400; mutually exclusive with other time elements)</p> <p>U (1 - 12 times) Fractional seconds (number of occurrences of U represent the number of digits with each digit in the range 0 - 9)</p> <p>TT Meridian indicator (AM or PM)</p> <p>Following is an example of a time stamp format:</p> <p>"YYYY/MM/DD HH:MM:SS.UUUUUU"</p> <p>The MMM element produces the following values: 'Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', and 'Dec'. 'Jan' is equal to month 1, and 'Dec' is equal to month 12.</p> <p>The following example illustrates how to export data that contains user-defined time stamp formats from a table called 'schedule':</p> <pre>db2 export to delfile2 of del modified by timestampformat="yyyy.mm.dd hh:mm tt" select * from schedule</pre>

Table 5. Valid file type modifiers for the export utility: IXF file format

Modifier	Description
codepage=x	<p>x is an ASCII character string. The value is interpreted as the code page of the data in the output data set. Converts character data from this code page to the application code page during the export operation.</p> <p>For pure DBCS (graphic), mixed DBCS, and EUC, delimiters are restricted to the range of x00 to x3F, inclusive.</p>

Note:

1. The export utility does not issue a warning if an attempt is made to use unsupported file types with the **MODIFIED BY** option. If this step is attempted, the export operation fails, and an error code is returned.
2. *Delimiter considerations for moving data* lists restrictions that apply to the characters that can be used as delimiter overrides.
3. The export utility normally writes
 - date data in YYYYMMDD format
 - char(date) data in "YYYY-MM-DD" format
 - time data in "HH.MM.SS" format
 - time stamp data in "YYYY-MM-DD-HH.MM.SS.aaaaaa" format

Data that is contained in any datetime columns that are specified in the SELECT statement for the export operation are also in these formats.

4. For time stamp formats, care must be taken to avoid ambiguity between the month and the minute descriptors, since they both use the letter M. A month field must be next to other date fields. A minute field must be next to other time fields. Following are some ambiguous time stamp formats:

"M" (could be a month, or a minute)
"M:M" (Which is which?)
"M:YYYY:M" (Both are interpreted as month.)
"S:M:YYYY" (adjacent to both a time value and a date value)

In ambiguous cases, the utility reports an error message, and the operation fails.

Following are some unambiguous time stamp formats:

"M:YYYY" (Month)
"S:M" (Minute)
"M:YYYY:S:M" (Month...Minute)
"M:H:YYYY:M:D" (Minute...Month)

5. All XDM instances are written to XML files that are separate from the main data file, even if the **XMLFILE** and **XML TO** clauses are not specified. By default, XML files are written to the path of the exported data file. The default base name for XML files is the name of the exported data file with the extension ".XML" appended to it.
6. All XDM instances are written with an XML declaration at the beginning that includes an encoding attribute, unless the XMLNODEDECLARATION file type modifier is specified.
7. By default, all XDM instances are written in Unicode unless the XMLCHAR or XMLGRAPHIC file type modifier is specified.
8. The default path for XML data and LOB data is the path of the main data file. The default XML file base name is the main data file. The default LOB file base name is the main data file. For example, if the main data file is:

/mypath/myfile.del

The default path for XML data and LOB data is:

/mypath"

The default XML file base name is:

myfile.del

The default LOB file base name is:

`myfile.del`

The LOBSINFILE file type modifier must be specified in order to have LOB files generated.

9. The export utility appends a numeric identifier to each LOB file or XML file. The identifier starts as a 3-digit, 0 padded sequence value, starting at:
`.001`

After the 999th LOB file or XML file, the identifier will no longer be padded with zeros (for example, the 1000th LOG file or XML file will have an extension of:

`.1000`

Following the numeric identifier is a three character type identifier that represents the data type, either:

`.lob`

or

`.xml`

For example, a generated LOB file would have a name in the format:

`myfile.del.001.lob`

and a generated XML file would have a name in the format:

`myfile.del.001.xml`

10. It is possible to have the export utility export XDM instances that are not well-formed documents by specifying an XQuery. However, you cannot import or load these exported documents directly into an XML column, since XML columns can contain only complete documents.

FORCE APPLICATION command using the ADMIN_CMD procedure:

Forces local or remote users or applications off the system to allow for maintenance on a server.

Attention: If an operation that cannot be interrupted (**RESTORE DATABASE**, for example) is forced, the operation must be successfully re-executed before the database becomes available.

Scope

This command affects all database partitions that are listed in the `$HOME/sql11ib/db2nodes.cfg` file.

In a partitioned database environment, this command does not have to be issued from the coordinator database partition of the application being forced. It can be issued from any database partition server in the partitioned database environment.

Authorization

One of the following authorities:

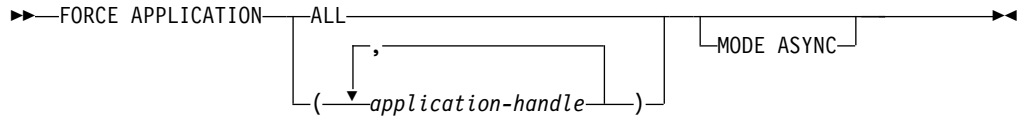
- SYSADM
- SYSCTRL

- SYSMANT

Required connection

Instance. To force users off a remote server, it is first necessary to attach to that server. If no attachment exists, this command is executed locally.

Command syntax



Command parameters

FORCE APPLICATION

ALL All applications will be disconnected from the database server.

application-handle

Specifies the agent to be terminated. List the values using the **LIST APPLICATIONS** command.

MODE ASYNC

The command does not wait for all specified users to be terminated before returning; it returns as soon as the function has been successfully issued or an error (such as invalid syntax) is discovered.

This is the only mode that is currently supported.

Examples

The following example forces two users, with *application-handle* values of 41408 and 55458, to disconnect from the database:

```
db2 "force application ( 41408, 55458 )"
```

Usage notes

The database manager remains active so that subsequent database manager operations can be handled without the need for **db2start**.

To preserve database integrity, only users who are idling or executing interruptible database operations can be terminated.

The following types of users and applications cannot be forced:

- users creating a database
- system applications

In order to successfully force these types of users and applications, the database must be deactivated and/or the instance restarted.

After a **FORCE APPLICATION** has been issued, the database will still accept requests to connect. Additional forces might be required to completely force all users off.

GET STMM TUNING command using the ADMIN_CMD procedure:

Used to read the catalog tables to report the user preferred self tuning memory manager (STMM) tuning member number and current STMM tuning member number.

Authorization

The privileges held by the authorization ID of the statement must include at least one of the following authorities or privilege:

- DBADM
- SECADM
- SQLADM
- ACCESSCTRL
- DATAACCESS
- SELECT on SYSIBM.SYSTUNINGINFO

Required connection

Database

Command syntax

►►—GET—STMM—TUNING—MEMBER—◄◄

Example

```
CALL SYSPROC.ADMIN_CMD( 'get stmm tuning member' )
```

The following is an example of output from this query.

Result set 1

```
-----  
USER_PREFERRED_NUMBER CURRENT_NUMBER  
-----  
2 2
```

1 record(s) selected.

Return Status = 0

Usage notes

- The user preferred self tuning memory manager (STMM) tuning member number (USER_PREFERRED_NUMBER) is set by the user and specifies the member on which the user wants to run the memory tuner. While the database is running, the tuning member is applied a few times an hour. As a result, it is possible that the CURRENT_NUMBER and USER_PREFERRED_NUMBER returned are not in sync after an update of the user preferred STMM member. To resolve this, either wait for the CURRENT_NUMBER to be updated asynchronously, or stop and start the database to force the update of CURRENT_NUMBER.

Compatibilities

For compatibility with previous versions:

- **DBPARTITIONNUM** can be substituted for **MEMBER**, except when the **DB2_ENFORCE_MEMBER_SYNTAX** registry variable is set to ON.

IMPORT command using the ADMIN_CMD procedure:

Inserts data from an external file with a supported file format into a table, hierarchy, view or nickname. **LOAD** is a faster alternative, but the load utility does not support loading data at the hierarchy level.

Quick link to “File type modifiers for the import utility” on page 64.

Authorization

- **IMPORT** using the **INSERT** option requires one of the following authorities:
 - DATAACCESS authority
 - CONTROL privilege on each participating table, view, or nickname
 - INSERT and SELECT privilege on each participating table or view
- **IMPORT** to an existing table using the **INSERT_UPDATE** option, requires one of the following authorities:
 - DATAACCESS authority
 - CONTROL privilege on each participating table, view, or nickname
 - INSERT, SELECT, UPDATE and DELETE privilege on each participating table or view
- **IMPORT** to an existing table using the **REPLACE** or **REPLACE_CREATE** option, requires one of the following authorities:
 - DATAACCESS authority
 - CONTROL privilege on the table or view
 - INSERT, SELECT, and DELETE privilege on the table or view
- **IMPORT** to a new table using the **CREATE** or **REPLACE_CREATE** option, requires one of the following authorities:
 - DBADM authority
 - CREATETAB authority on the database and USE privilege on the table space, as well as one of:
 - IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the table does not exist
 - CREATEIN privilege on the schema, if the schema name of the table refers to an existing schema
- **IMPORT** to a hierarchy that does not exist using the **CREATE**, or the **REPLACE_CREATE** option, requires one of the following authorities:
 - DBADM authority
 - CREATETAB authority on the database and USE privilege on the table space and one of:
 - IMPLICIT_SCHEMA authority on the database, if the schema name of the table does not exist
 - CREATEIN privilege on the schema, if the schema of the table exists
 - CONTROL privilege on every sub-table in the hierarchy, if the **REPLACE_CREATE** option on the entire hierarchy is used
- **IMPORT** to an existing hierarchy using the **REPLACE** option requires one of the following authorities:
 - DATAACCESS authority
 - CONTROL privilege on every sub-table in the hierarchy

- To import data into a table that has protected columns, the session authorization ID must have LBAC credentials that allow write access to all protected columns in the table. Otherwise the import fails and an error (SQLSTATE 42512) is returned.
- To import data into a table that has protected rows, the session authorization ID must hold LBAC credentials that meet these criteria:
 - It is part of the security policy protecting the table
 - It was granted to the session authorization ID for write access

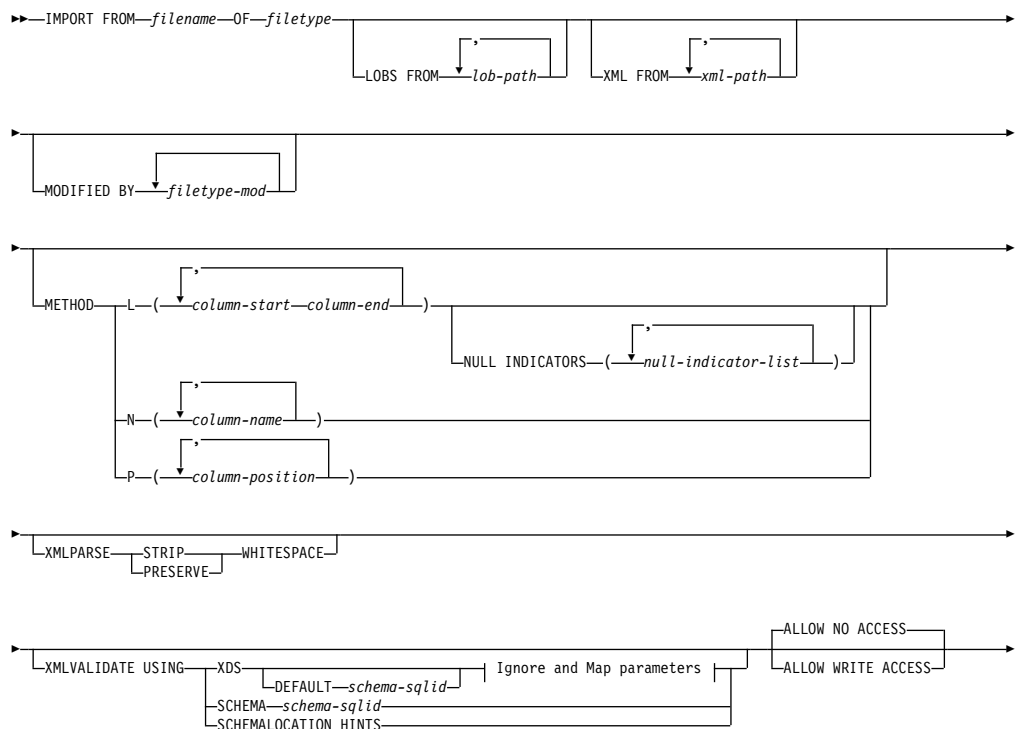
The label on the row to insert, the user's LBAC credentials, the security policy definition, and the LBAC rules determine the label on the row.

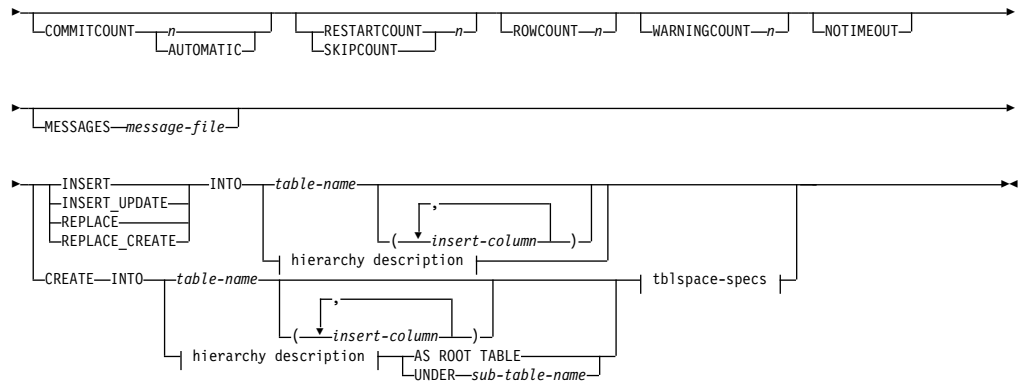
- For a table with protected rows, if the **REPLACE** or **REPLACE_CREATE** option is specified, the session authorization ID must have the authority to drop the table.
- To import data into a nickname, the session authorization ID must have the privilege to access and use a specified data source in pass-through mode.
- If the table has row access control activated, then **IMPORT REPLACE** on that table would require the ability to drop the table. Specifically, you must have either **CONTROL** or **DBADM** on the table.

Required connection

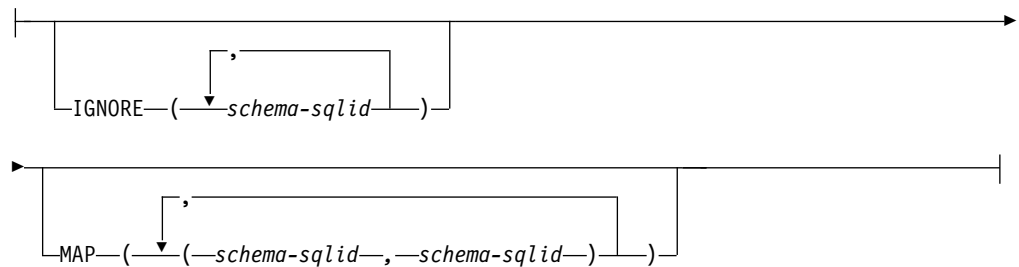
Database. If implicit connect is enabled, a connection to the default database is established. Utility access to Linux, UNIX, or Windows database servers from Linux, UNIX, or Windows clients must be a direct connection through the engine and not through a Db2 Connect gateway or loop back environment.

Command syntax





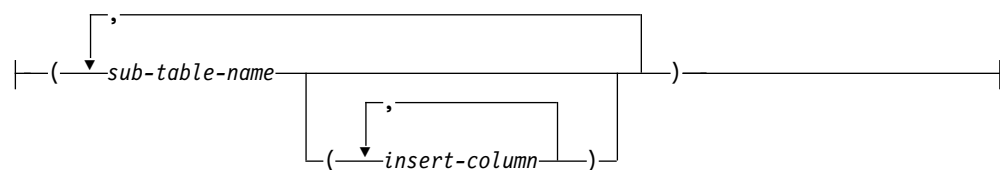
Ignore and Map parameters:



hierarchy description:



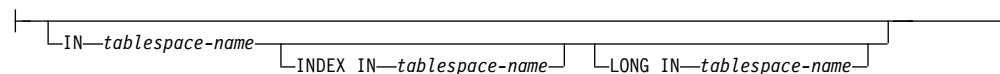
sub-table-list:



traversal-order-list:



tblspace-specs:



Command parameters

FROM *filename*

Specifies the file that contains the data to be imported. If the path is omitted, the current working directory is used.

OF *filetype*

Specifies the format of the data in the input file:

- ASC (non-delimited ASCII format)
- DEL (delimited ASCII format), which is used by a variety of database manager and file manager programs
- IXF (Integration Exchange Format, PC version) is a binary format that is used exclusively by Db2.

LOBS FROM *lob-path*

Specifies one or more paths that store LOB files. The names of the LOB data files are stored in the main data file (ASC, DEL, or IXF), in the column that will be loaded into the LOB column. The maximum number of paths that can be specified is 999. This will implicitly activate the LOBSINFILE behavior.

This parameter is not valid when you import to a nickname.

XML FROM *xml-path*

Specifies one or more paths that contain the XML files.

MODIFIED BY *filetype-mod*

Specifies file type modifier options. See “File type modifiers for the import utility” on page 64.

METHOD

- L** Specifies the start and end column numbers from which to import data. A column number is a byte offset from the beginning of a row of data. It is numbered starting from 1.

Note: This method can only be used with ASC files, and is the only valid option for that file type.

- N** Specifies the names of the columns in the data file to be imported. The case of these column names must match the case of the corresponding names in the system catalogs. Each table column that is not nullable should have a corresponding entry in the **METHOD N** list. For example, given data fields F1, F2, F3, F4, F5, and F6, and table columns C1 INT, C2 INT NOT NULL, C3 INT NOT NULL, and C4 INT, method N (F2, F1, F4, F3) is a valid request, while method N (F2, F1) is not valid.

Note: This method can only be used with IXF files.

- P** Specifies the field numbers (numbered from 1) of the input data fields to be imported. Each table column that is not nullable should have a corresponding entry in the **METHOD P** list. For example, given data fields F1, F2, F3, F4, F5, and F6, and table columns C1 INT, C2 INT NOT NULL, C3 INT NOT NULL, and C4 INT, method P (2, 1, 4, 3) is a valid request, while method P (2, 1) is not valid. This method can only be used with file types IXF or DEL, and is the only valid method for the DEL file type.

For each of the fields specified by method P, you need to define a corresponding column in the action statement, unless all columns are accounted for or the first *x* columns are going to be loaded, as shown in the following example:

```
db2 load from datafile1.del of del method P(1, 3, 4)
      replace into table1 (c1, c3, c4)
```

NULL INDICATORS *null-indicator-list*

This option can only be used when the **METHOD L** parameter is specified. That is, the input file is an ASC file. The null indicator list is a comma-separated list of positive integers specifying the column number of each null indicator field. The column number is the byte offset of the null indicator field from the beginning of a row of data. There must be one entry in the null indicator list for each data field defined in the **METHOD L** parameter. A column number of zero indicates that the corresponding data field always contains data.

A value of Y in the NULL indicator column specifies that the column data is NULL. Any character *other than* Y in the NULL indicator column specifies that the column data is not NULL, and that column data specified by the **METHOD L** option will be imported.

The NULL indicator character can be changed using the **MODIFIED BY** option, with the nullindchar file type modifier.

XMLPARSE

Specifies how XML documents are parsed. If this option is not specified, the parsing behavior for XML documents will be determined by the value of the CURRENT IMPLICIT XMLPARSE OPTION special register.

STRIP WHITESPACE

Specifies to remove whitespace when the XML document is parsed.

PRESERVE WHITESPACE

Specifies not to remove whitespace when the XML document is parsed.

XMLVALIDATE

Specifies that XML documents are validated against a schema, when applicable.

USING XDS

XML documents are validated against the XML schema identified by the XML Data Specifier (XDS) in the main data file. By default, if the **XMLVALIDATE** option is invoked with the **USING XDS** clause, the schema used to perform validation will be determined by the SCH attribute of the XDS. If an SCH attribute is not present in the XDS, no schema validation will occur unless a default schema is specified by the **DEFAULT** clause.

The **DEFAULT**, **IGNORE**, and **MAP** clauses can be used to modify the schema determination behavior. These three optional clauses apply directly to the specifications of the XDS, and not to each other. For example, if a schema is selected because it is specified by the **DEFAULT** clause, it will not be ignored if also specified by the **IGNORE** clause. Similarly, if a schema is selected because it is specified as the first part of a pair in the MAP clause, it will not be re-mapped if also specified in the second part of another **MAP** clause pair.

USING SCHEMA *schema-sqlid*

XML documents are validated against the XML schema with the

specified SQL identifier. In this case, the SCH attribute of the XML Data Specifier (XDS) will be ignored for all XML columns.

USING SCHEMALOCATION HINTS

XML documents are validated against the schemas identified by XML schema location hints in the source XML documents. If a schemaLocation attribute is not found in the XML document, no validation will occur. When the **USING SCHEMALOCATION HINTS** clause is specified, the SCH attribute of the XML Data Specifier (XDS) will be ignored for all XML columns.

See examples of the **XMLVALIDATE** option in the following section.

DEFAULT *schema-sqlid*

This option can only be used when the **USING XDS** parameter is specified. The schema specified through the **DEFAULT** clause identifies a schema to use for validation when the XML Data Specifier (XDS) of an imported XML document does not contain an SCH attribute identifying an XML Schema.

The **DEFAULT** clause takes precedence over the **IGNORE** and **MAP** clauses. If an XDS satisfies the **DEFAULT** clause, the **IGNORE** and **MAP** specifications will be ignored.

IGNORE *schema-sqlid*

This option can only be used when the **USING XDS** parameter is specified. The **IGNORE** clause specifies a list of one or more schemas to ignore if they are identified by an SCH attribute. If an SCH attribute exists in the XML Data Specifier for an imported XML document, and the schema identified by the SCH attribute is included in the list of schemas to ignore, then no schema validation will occur for the imported XML document.

If a schema is specified in the **IGNORE** clause, it cannot also be present in the left side of a schema pair in the **MAP** clause.

The **IGNORE** clause applies only to the XDS. A schema that is mapped by the **MAP** clause will not be subsequently ignored if specified by the **IGNORE** clause.

MAP *schema-sqlid*

This option can only be used when the **USING XDS** parameter is specified. Use the **MAP** clause to specify alternate schemas to use in place of those specified by the SCH attribute of an XML Data Specifier (XDS) for each imported XML document. The **MAP** clause specifies a list of one or more schema pairs, where each pair represents a mapping of one schema to another. The first schema in the pair represents a schema that is referred to by an SCH attribute in an XDS. The second schema in the pair represents the schema that should be used to perform schema validation.

If a schema is present in the left side of a schema pair in the **MAP** clause, it cannot also be specified in the **IGNORE** clause.

Once a schema pair mapping is applied, the result is final. The mapping operation is non-transitive, and therefore the schema chosen will not be subsequently applied to another schema pair mapping.

A schema cannot be mapped more than once, meaning that it cannot appear on the left side of more than one pair.

ALLOW NO ACCESS

Runs import in the offline mode. An exclusive (X) lock on the target table

is acquired before any rows are inserted. This prevents concurrent applications from accessing table data. This is the default import behavior.

ALLOW WRITE ACCESS

Runs import in the online mode. An intent exclusive (IX) lock on the target table is acquired when the first row is inserted. This allows concurrent readers and writers to access table data. Online mode is not compatible with the **REPLACE**, **CREATE**, or **REPLACE_CREATE** import options. Online mode is not supported in conjunction with buffered inserts. The import operation will periodically commit inserted data to prevent lock escalation to a table lock and to avoid running out of active log space. These commits will be performed even if the **COMMITCOUNT** option was not used. During each commit, import will lose its IX table lock, and will attempt to reacquire it after the commit. This parameter is required when you import to a nickname and **COMMITCOUNT** must be specified with a valid number (AUTOMATIC is not considered a valid option).

COMMITCOUNT *n* | AUTOMATIC

Performs a COMMIT after every *n* records are imported. When a number *n* is specified, import performs a COMMIT after every *n* records are imported. When compound inserts are used, a user-specified commit frequency of *n* is rounded up to the first integer multiple of the compound count value. When AUTOMATIC is specified, import internally determines when a commit needs to be performed. The utility will commit for either one of two reasons:

- to avoid running out of active log space
- to avoid lock escalation from row level to table level

If the **ALLOW WRITE ACCESS** option is specified, and the **COMMITCOUNT** option is not specified, the import utility will perform commits as if **COMMITCOUNT AUTOMATIC** had been specified.

The ability of the import operation to avoid running out of active log space is affected by the Db2 registry variable **DB2_FORCE_APP_ON_MAX_LOG**:

- If **DB2_FORCE_APP_ON_MAX_LOG** is set to FALSE and the **COMMITCOUNT AUTOMATIC** command option is specified, the import utility will be able to automatically avoid running out of active log space.
- If **DB2_FORCE_APP_ON_MAX_LOG** is set to FALSE and the **COMMITCOUNT *n*** command option is specified, the import utility will attempt to resolve the log full condition if it encounters an SQL0964C (Transaction Log Full) while inserting or updating a record. It will perform an unconditional commit and then will reattempt to insert or update the record. If this does not help resolve the issue (which would be the case when the log full is attributed to other activity on the database), then the **IMPORT** command will fail as expected, however the number of rows committed may not be a multiple of the **COMMITCOUNT *n*** value. To avoid processing the rows that were already committed when you retry the import operation, use the **RESTARTCOUNT** or **SKIPCOUNT** command parameters.
- If **DB2_FORCE_APP_ON_MAX_LOG** is set to TRUE (which is the default), the import operation will fail if it encounters an SQL0964C while inserting or updating a record. This can occur irrespective of whether you specify **COMMITCOUNT AUTOMATIC** or **COMMITCOUNT *n***.

The application is forced off the database and the current unit of work is rolled back. To avoid processing the rows that were already committed when you retry the import operation, use the **RESTARTCOUNT** or **SKIPCOUNT** command parameters.

RESTARTCOUNT *n*

Specifies that an import operation is to be started at record $n+1$. The first n records are skipped. This option is functionally equivalent to **SKIPCOUNT**. **RESTARTCOUNT** and **SKIPCOUNT** are mutually exclusive.

SKIPCOUNT *n*

Specifies that an import operation is to be started at record $n+1$. The first n records are skipped. This option is functionally equivalent to **RESTARTCOUNT**. **SKIPCOUNT** and **RESTARTCOUNT** are mutually exclusive.

ROWCOUNT *n*

Specifies the number n of physical records in the file to be imported (inserted or updated). Allows a user to import only n rows from a file, starting from the record determined by the **SKIPCOUNT** or **RESTARTCOUNT** options. If the **SKIPCOUNT** or **RESTARTCOUNT** options are not specified, the first n rows are imported. If **SKIPCOUNT *m*** or **RESTARTCOUNT *m*** is specified, rows $m+1$ to $m+n$ are imported. When compound inserts are used, user specified **ROWCOUNT *n*** is rounded up to the first integer multiple of the compound count value.

WARNINGCOUNT *n*

Stops the import operation after n warnings. Set this parameter if no warnings are expected, but verification that the correct file and table are being used is required. If the import file or the target table is specified incorrectly, the import utility will generate a warning for each row that it attempts to import, which will cause the import to fail. If n is zero, or this option is not specified, the import operation will continue regardless of the number of warnings issued.

NOTIMEOUT

Specifies that the import utility will not time out while waiting for locks. This option supersedes the **locktimeout** database configuration parameter. Other applications are not affected.

MESSAGES *message-file*

Specifies the destination for warning and error messages that occur during an import operation. If the file already exists, the import utility appends the information. If the complete path to the file is not specified, the utility uses the current directory and the default drive as the destination. If *message-file* is omitted, the messages are written to standard output.

INSERT Adds the imported data to the table without changing the existing table data.

INSERT_UPDATE

Adds rows of imported data to the target table, or updates existing rows (of the target table) with matching primary keys.

REPLACE

Deletes all existing data from the table by truncating the data object, and inserts the imported data. The table definition and the index definitions are not changed. This option can only be used if the table exists. If this option is used when moving data between hierarchies, only the data for an entire hierarchy, not individual subtables, can be replaced.

This parameter is not valid when you import to a nickname.

This option does not honor the CREATE TABLE statement's NOT LOGGED INITIALLY (NLI) clause or the ALTER TABLE statement's ACTIVE NOT LOGGED INITIALLY clause.

This option cannot be used to import data into system-period temporal tables.

If an import with the **REPLACE** option is performed within the same transaction as a CREATE TABLE or ALTER TABLE statement where the NLI clause is invoked, the import will not honor the NLI clause. All inserts will be logged.

Workaround 1

Delete the contents of the table using the DELETE statement, then invoke the import with INSERT statement

Workaround 2

Drop the table and re-create it, then invoke the import with INSERT statement.

REPLACE_CREATE

Note: The **REPLACE_CREATE** parameter is deprecated and may be removed in a future release. For additional details, see "IMPORT command options CREATE and REPLACE_CREATE are deprecated".

If the table exists, deletes all existing data from the table by truncating the data object, and inserts the imported data without changing the table definition or the index definitions.

If the table does not exist, creates the table and index definitions, as well as the row contents, in the code page of the database. See *Imported table re-creation* for a list of restrictions.

This option can only be used with IXF files. If this option is used when moving data between hierarchies, only the data for an entire hierarchy, not individual subtables, can be replaced.

This parameter is not valid when you import to a nickname.

INTO *table-name*

Specifies the database table into which the data is to be imported. This table cannot be a system table, a created temporary table, a declared temporary table, or a summary table.

One can use an alias for **INSERT**, **INSERT_UPDATE**, or **REPLACE**, except in the case of an earlier server, when the fully qualified or the unqualified table name should be used. A qualified table name is in the form: *schema.tablename*. The *schema* is the user name under which the table was created.

If the database table contains implicitly hidden columns, you must specify whether data for the hidden columns is included in the import operation. Use one of the following methods to indicate if data for hidden columns is included:

- Use *insert-column* to explicitly specify the columns into which data is to be inserted.

```
db2 import from delfile1 of del
insert into table1 (c1, c2, c3,...)
```

- Use one of the hidden column file type modifiers: specify **implicitlyhiddeninclude** when the input file contains data for the hidden columns, or **implicitlyhiddenmissing** when the input file does not.

```
db2 import from delfile1 of del modified by implicitlyhiddeninclude
insert into table1
```

- Use the DB2_DMU_DEFAULT registry variable on the client-side to set the default behavior when data movement utilities encounter tables with implicitly hidden columns.

```
db2set DB2_DMU_DEFAULT=IMPLICITLYHIDDENINCLUDE
db2 import from delfile1 of del insert into table1
```

insert-column

Specifies the name of a column in the table or the view into which data is to be inserted.

ALL TABLES

An implicit keyword for hierarchy only. When importing a hierarchy, the default is to import all tables specified in the traversal order.

sub-table-list

For typed tables with the **INSERT** or the **INSERT_UPDATE** option, a list of sub-table names is used to indicate the sub-tables into which data is to be imported.

HIERARCHY

Specifies that hierarchical data is to be imported.

STARTING *sub-table-name*

A keyword for hierarchy only, requesting the default order, starting from *sub-table-name*. For PC/IXF files, the default order is the order stored in the input file. The default order is the only valid order for the PC/IXF file format.

traversal-order-list

For typed tables with the **INSERT**, **INSERT_UPDATE**, or the **REPLACE** option, a list of sub-table names is used to indicate the traversal order of the importing sub-tables in the hierarchy.

CREATE

Note: The **CREATE** parameter is deprecated and may be removed in a future release. For additional details, see “IMPORT command options **CREATE** and **REPLACE_CREATE** are deprecated”.

Creates the table definition and row contents in the code page of the database. If the data was exported from a Db2 table, sub-table, or hierarchy, indexes are created. If this option operates on a hierarchy, and data was exported from Db2, a type hierarchy will also be created. This option can only be used with IXF files.

This parameter is not valid when you import to a nickname.

Note: If the data was exported from an MVS™ host database, and it contains LONGVAR fields whose lengths, calculated on the page size, are more than 254, **CREATE** might fail because the rows are too long. See “Imported table re-creation” for a list of restrictions. In this case, the table should be created manually, and **IMPORT** with **INSERT** should be invoked, or, alternatively, the **LOAD** command should be used.

AS ROOT TABLE

Creates one or more sub-tables as a stand-alone table hierarchy.

UNDER *sub-table-name*

Specifies a parent table for creating one or more sub-tables.

IN *tablespace-name*

Identifies the table space in which the table will be created. The table space must exist, and must be a REGULAR table space or LARGE table space . If no other table space is specified, all table parts are stored in this table space. If this clause is not specified, the table is created in a table space created by the authorization ID. If none is found, the table is placed into the default table space USERSPACE1. If USERSPACE1 has been dropped, table creation fails.

INDEX IN *tablespace-name*

Identifies the table space in which any indexes on the table will be created. This option is allowed only when the primary table space specified in the **IN** clause is a DMS table space. The specified table space must exist, and must be a REGULAR or LARGE DMS table space.

Note: Specifying which table space will contain an index can only be done when the table is created.

LONG IN *tablespace-name*

Identifies the table space in which the values of any long columns (LONG VARCHAR, LONG VARGRAPHIC, LOB data types, or distinct types with any of these as source types) will be stored. This option is allowed only if the primary table space specified in the **IN** clause is a DMS table space. The table space must exist, and must be a LARGE DMS table space.

Examples**Example 1**

The following example shows how to import information from myfile.ixf to the STAFF table:

```
db2 import from myfile.ixf of ixf messages msg.txt insert into staff

SQL3150N  The H record in the PC/IXF file has product "DB2    01.00", date
"19970220", and time "140848".

SQL3153N  The T record in the PC/IXF file has name "myfile",
qualifier "      ", and source "      ".

SQL3109N  The utility is beginning to load data from file "myfile".

SQL3110N  The utility has completed processing.  "58" rows were read
from the input file.

SQL3221W  ...Begin COMMIT WORK. Input Record Count = "58".

SQL3222W  ...COMMIT of any database changes was successful.

SQL3149N  "58" rows were processed from the input file.  "58" rows were
successfully inserted into the table.  "0" rows were rejected.
```

Example 2 (Importing into a table with an identity column)

TABLE1 has 4 columns:

- C1 VARCHAR(30)

- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 is the same as TABLE1, except that C2 is a GENERATED ALWAYS identity column.

Data records in DATAFILE1 (DEL format):

```
"Liszt"
"Hummel",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

Data records in DATAFILE2 (DEL format):

```
"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

The following command generates identity values for rows 1 and 2, since no identity values are supplied in DATAFILE1 for those rows. However, row 3 is assigned the user-supplied identity value of 100 and row 4 is assigned the the user-supplied identity value of 101.

```
db2 import from datafile1.del of del replace into table1
```

To import DATAFILE1 into TABLE1 so that identity values are generated for all rows, issue one of the following commands:

```
db2 import from datafile1.del of del method P(1, 3, 4)
  replace into table1 (c1, c3, c4)
db2 import from datafile1.del of del modified by identityignore
  replace into table1
```

To import DATAFILE2 into TABLE1 so that identity values are generated for each row, issue one of the following commands:

```
db2 import from datafile2.del of del replace into table1 (c1, c3, c4)
db2 import from datafile2.del of del modified by identitymissing
  replace into table1
```

If DATAFILE1 is imported into TABLE2 without using any of the identity-related file type modifiers, rows 1 and 2 will be inserted, but rows 3 and 4 will be rejected, because they supply their own non-NULL values, and the identity column is GENERATED ALWAYS.

Examples of using the XMLVALIDATE clause

Example 1 (XMLVALIDATE USING XDS)

For the following XMLVALIDATE clause:

```
XMLVALIDATE USING XDS
  IGNORE (S1.SCHEMA_A)
  MAP ((S1.SCHEMA_A, S2.SCHEMA_B))
```

The import would fail due to invalid syntax, since the **IGNORE** of S1.SCHEMA_A would conflict with the **MAP** of S1.SCHEMA_A to S2.SCHEMA_B.

Example 2 (XMLVALIDATE USING XDS)

For the following **XMLVALIDATE** clause:

```
XMLVALIDATE USING XDS
  DEFAULT S8.SCHEMA_H
  IGNORE (S9.SCHEMA_I, S10.SCHEMA_J)
  MAP ((S1.SCHEMA_A, S2.SCHEMA_B), (S3.SCHEMA_C, S5.SCHEMA_E),
       (S6.SCHEMA_F, S3.SCHEMA_C), (S4.SCHEMA_D, S7.SCHEMA_G))
```

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The XML schema with SQL identifier "S8.SCHEMA_H" is used to validate the document in file "xmlfile.001.xml", since "S8.SCHEMA_H" was specified as the default schema to use.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' OFF='10' LEN='500' SCH='S10.SCHEMA_J' />
```

No schema validation occurs for the document in file "xmlfile.002.xml", since although the XDS specifies "S10.SCHEMA_J" as the schema to use, that schema is part of the **IGNORE** clause. The document contents can be found at byte offset 10 in the file (meaning the 11th byte), and is 500 bytes long.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.003.xml' SCH='S6.SCHEMA_F' />
```

The XML schema with SQL identifier "S3.SCHEMA_C" is used to validate the document in file "xmlfile.003.xml". This is because the **MAP** clause specifies that schema "S6.SCHEMA_F" should be mapped to schema "S3.SCHEMA_C". Note that further mapping does not take place, therefore the mapping of schema "S3.SCHEMA_C" to schema "S5.SCHEMA_E" does not apply in this case.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.004.xml' SCH='S11.SCHEMA_K' />
```

The XML schema with SQL identifier "S11.SCHEMA_K" is used to validate the document in file "xmlfile.004.xml". Note that none of the **DEFAULT**, **IGNORE**, or **MAP** specifications apply in this case.

Example 3 (XMLVALIDATE USING XDS)

For the following **XMLVALIDATE** clause:

```
XMLVALIDATE USING XDS
  DEFAULT S1.SCHEMA_A
  IGNORE (S1.SCHEMA_A)
```

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The XML schema with SQL identifier "S1.SCHEMA_A" is used to validate the document in file "xmlfile.001.xml", since "S1.SCHEMA_1" was specified as the default schema to use.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' SCH='S1.SCHEMA_A' />
```

No schema validation occurs for the document in file "xmlfile.002", since although the XDS specifies "S1.SCHEMA_A" as the schema to use, that schema is part of the **IGNORE** clause.

Example 4 (XMLVALIDATE USING XDS)

For the following **XMLVALIDATE** clause:

```
XMLVALIDATE USING XDS
  DEFAULT S1.SCHEMA_A
  MAP ((S1.SCHEMA_A, S2.SCHEMA_B), (S2.SCHEMA_B, S1.SCHEMA_A))
```

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The XML schema with SQL identifier "S1.SCHEMA_A" is used to validate the document in file "xmlfile.001.xml", since "S1.SCHEMA_1" was specified as the default schema to use. Note that since the **DEFAULT** clause was applied, the **MAP** clause is not subsequently applied. Therefore the mapping of schema "S1.SCHEMA_A" to schema "S2.SCHEMA_B" does not apply in this case.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' SCH='S1.SCHEMA_A' />
```

The XML schema with SQL identifier "S2.SCHEMA_B" is used to validate the document in file "xmlfile.002.xml". This is because the **MAP** clause specifies that schema "S1.SCHEMA_A" should be mapped to schema "S2.SCHEMA_B". Note that further mapping does not take place, therefore the mapping of schema "S2.SCHEMA_B" to schema "S1.SCHEMA_A" does not apply in this case.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.003.xml' SCH='S2.SCHEMA_B' />
```

The XML schema with SQL identifier "S1.SCHEMA_A" is used to validate the document in file "xmlfile.003.xml". This is because the **MAP** clause specifies that schema "S2.SCHEMA_B" should be mapped to schema "S1.SCHEMA_A". Note that further mapping does not take place, therefore the mapping of schema "S1.SCHEMA_A" to schema "S2.SCHEMA_B" does not apply in this case.

Example 5 (XMLVALIDATE USING SCHEMA)

For the following **XMLVALIDATE** clause:

```
XMLVALIDATE USING SCHEMA S2.SCHEMA_B
```

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The document in file xmlfile.001.xml is validated using the XML schema with SQL identifier "S2.SCHEMA_B".

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' SCH='S1.SCHEMA_A' />
```

The document in file "xmlfile.002.xml" is validated using the XML schema with SQL identifier "S2.SCHEMA_B". Note that the SCH attribute is ignored, since validation is being performed using a schema specified by the **USING SCHEMA** clause.

Example 6 (XMLVALIDATE USING SCHEMALOCATION HINTS)

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The XML schema used is determined by the schemaLocation attribute in the document contents, and no validation would occur if one is not present.

For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' SCH='S1.SCHEMA_A' />
```

The XML schema used is determined by the schemaLocation attribute in the document contents, and no validation would occur if one is not present. Note that the SCH attribute is ignored, since validation is being performed using **SCHEMALOCATION HINTS**.

Usage notes

Be sure to complete all table operations and release all locks before starting an import operation. This can be done by issuing a COMMIT after closing all cursors opened WITH HOLD, or by issuing a ROLLBACK.

The import utility adds rows to the target table using the SQL INSERT statement. The utility issues one INSERT statement for each row of data in the input file. If an INSERT statement fails, one of two actions result:

- If it is likely that subsequent INSERT statements can be successful, a warning message is written to the message file, and processing continues.
- If it is likely that subsequent INSERT statements will fail, and there is potential for database damage, an error message is written to the message file, and processing halts.

The utility performs an automatic COMMIT after the old rows are deleted during a **REPLACE** or a **REPLACE_CREATE** operation. Therefore, if the system fails, or the application interrupts the database manager after the table object is truncated, all of the old data is lost. Ensure that the old data is no longer needed before using these options.

If the log becomes full during a **CREATE**, **REPLACE**, or **REPLACE_CREATE** operation, the utility performs an automatic COMMIT on inserted records. If the system fails, or the application interrupts the database manager after an automatic COMMIT, a table with partial data remains in the database. Use the **REPLACE** or the **REPLACE_CREATE** option to rerun the whole import operation, or use **INSERT** with the **RESTARTCOUNT** parameter set to the number of rows successfully imported.

Updates from the IMPORT command will always be committed at the end of an IMPORT task. The IMPORT command can also perform automatic commits during its execution to reduce the size of the lock list and the active log space. The IMPORT command will roll back if the active log becomes full during IMPORT processing.

- By default, automatic commits are not performed for the **INSERT** or the **INSERT_UPDATE** option. They are, however, performed if the **COMMITCOUNT** parameter is not zero.
- Offline import does not perform automatic COMMITs if any of the following conditions are true:

- The target is a view, not a table
- Compound inserts are used
- Buffered inserts are used
- By default, online import performs automatic commit to free both the active log space and the lock list. Automatic commits are not performed only if a **COMMITCOUNT** value of zero is specified.

Whenever the import utility performs a COMMIT, two messages are written to the message file: one indicates the number of records to be committed, and the other is written after a successful COMMIT. When restarting the import operation after a failure, specify the number of records to skip, as determined from the last successful COMMIT.

The import utility accepts input data with minor incompatibility problems (for example, character data can be imported using padding or truncation, and numeric data can be imported with a different numeric data type), but data with major incompatibility problems is not accepted.

You cannot **REPLACE** or **REPLACE_CREATE** an object table if it has any dependents other than itself, or an object view if its base table has any dependents (including itself). To replace such a table or a view, do the following:

1. Drop all foreign keys in which the table is a parent.
2. Run the import utility.
3. Alter the table to re-create the foreign keys.

If an error occurs while recreating the foreign keys, modify the data to maintain referential integrity.

Referential constraints and foreign key definitions are not preserved when recreating tables from PC/IXF files. (Primary key definitions *are* preserved if the data was previously exported using SELECT *.)

Importing to a remote database requires enough disk space on the server for a copy of the input data file, the output message file, and potential growth in the size of the database.

If an import operation is run against a remote database, and the output message file is very long (more than 60 KB), the message file returned to the user on the client might be missing messages from the middle of the import operation. The first 30 KB of message information and the last 30 KB of message information are always retained.

Importing PC/IXF files to a remote database is much faster if the PC/IXF file is on a hard drive rather than on diskettes.

You cannot use the **IMPORT CREATE** option with a PC/IXF file format on a table that has an index defined with an expression-based key.

The database table or hierarchy must exist before data in the **ASC** or **DEL** file formats can be imported; however, if the table does not already exist, **IMPORT CREATE** or **IMPORT REPLACE_CREATE** creates the table when it imports data from a PC/IXF file. For typed tables, **IMPORT CREATE** can create the type hierarchy and the table hierarchy as well.

PC/IXF import should be used to move data (including hierarchical data) between databases. If character data containing row separators is exported to a delimited ASCII (DEL) file and processed by a text transfer program, fields containing the row separators will shrink or expand. The file copying step is not necessary if the source and the target databases are both accessible from the same client.

The data in ASC and DEL files is assumed to be in the code page of the client application performing the import. PC/IXF files, which allow for different code pages, are recommended when importing data in different code pages. If the PC/IXF file and the import utility are in the same code page, processing occurs as for a regular application. If the two differ, and the **FORCEIN** option is specified, the import utility assumes that data in the PC/IXF file has the same code page as the application performing the import. This occurs even if there is a conversion table for the two code pages. If the two differ, the **FORCEIN** option is not specified, and there is a conversion table, all data in the PC/IXF file will be converted from the file code page to the application code page. If the two differ, the **FORCEIN** option is not specified, and there is no conversion table, the import operation will fail. This applies only to PC/IXF files on Db2 clients on the AIX operating system.

For table objects on an 8 KB page that are close to the limit of 1012 columns, import of PC/IXF data files might cause Db2 to return an error, because the maximum size of an SQL statement was exceeded. This situation can occur only if the columns are of type CHAR, VARCHAR, or CLOB. The restriction does not apply to import of **DEL** or **ASC** files. If PC/IXF files are being used to create a new table, an alternative is use **db2look** to dump the DDL statement that created the table, and then to issue that statement through the CLP.

Db2 Connect can be used to import data to DRDA servers such as Db2 for z/OS, Db2 for VM and VSE, and Db2 for OS/400. Only PC/IXF import (**INSERT** option) is supported. The **RESTARTCOUNT** parameter, but not the **COMMITCOUNT** parameter, is also supported.

When using the **CREATE** option with typed tables, create every sub-table defined in the PC/IXF file; sub-table definitions cannot be altered. When using options other than **CREATE** with typed tables, the traversal order list enables one to specify the traverse order; therefore, the traversal order list must match the one used during the export operation. For the PC/IXF file format, one need only specify the target sub-table name, and use the traverse order stored in the file.

The import utility can be used to recover a table previously exported to a PC/IXF file. The table returns to the state it was in when exported.

Data cannot be imported to a system table, a created temporary table, a declared temporary table, or a summary table.

Views cannot be created through the import utility.

Importing a multiple-part PC/IXF file whose individual parts are copied from a Windows system to an AIX system is supported. Only the name of the first file must be specified in the **IMPORT** command. For example, **IMPORT FROM data.ixf OF IXF INSERT INTO TABLE1**. The file **data.002**, etc should be available in the same directory as **data.ixf**.

On the Windows operating system:

- Importing logically split PC/IXF files is not supported.

- Importing bad format PC/IXF files is not supported.

Security labels in their internal format might contain newline characters. If you import the file using the DEL file format, those newline characters can be mistaken for delimiters. If you have this problem use the older default priority for delimiters by specifying the `delprioritychar` file type modifier in the **IMPORT** command.

If the database table contains implicitly hidden columns, you must specify whether data for the hidden columns is included in the import operation.

The **IMPORT** utility does not match the number of columns in a table and the number of fields in a data file. The utility checks for a sufficient amount of data in the data file and if a row in the data file does not contain sufficient columns of data, the row may either be rejected with a warning message if the corresponding table columns without data are defined as NOT NULL, or be inserted successfully without a warning message if the corresponding table columns are defined as NULL. On the other hand, if a row contains a higher number of columns than required, the sufficient number of columns are processed while the remaining columns of data are omitted and no warning message is given.

Federated considerations

When using the **IMPORT** command and the **INSERT**, **UPDATE**, or **INSERT_UPDATE** command parameters, you must ensure that you have CONTROL privilege on the participating nickname. You must ensure that the nickname you want to use when doing an import operation already exists. There are also several restrictions you should be aware of as shown in the **IMPORT** command parameters section.

Some data sources, such as ODBC, do not support importing into nicknames.

Column-organized tables

- The **IMPORT** command inherits all INSERT, UPDATE, or DELETE statement restrictions that pertain to column-organized tables.
- The **CREATE INTO** parameter of the **IMPORT** command cannot create a column-organized table.
- Import operations against column-organized tables use the CS isolation level (which is needed for insert operations), not the default RS isolation level, which is set when import packages are bound to the database.
- You cannot use an IXF file that you created by exporting data from a column-organized table with the **CREATE** parameter or the **REPLACE_CREATE** parameter of the **IMPORT** command.

File type modifiers for the import utility

Table 6. Valid file type modifiers for the import utility: All file formats

Modifier	Description
compound= <i>x</i>	<p><i>x</i> is a number between 1 and 100 inclusive. Uses nonatomic compound SQL to insert the data, and <i>x</i> statements will be attempted each time.</p> <p>If this modifier is specified, and the transaction log is not sufficiently large, the import operation will fail. The transaction log must be large enough to accommodate either the number of rows specified by COMMITCOUNT, or the number of rows in the data file if COMMITCOUNT is not specified. It is therefore recommended that the COMMITCOUNT option be specified to avoid transaction log overflow.</p> <p>This modifier is incompatible with:</p> <ul style="list-style-type: none"> • INSERT_UPDATE mode • Hierarchical tables • Random distribution tables that use the random by generation method • The following modifiers: usedefaults, identitymissing, identityignore, generatedmissing, and generatedignore
generatedignore	<p>This modifier informs the import utility that data for all generated columns is present in the data file but should be ignored. This results in all values for the generated columns being generated by the utility. This modifier cannot be used with the generatedmissing modifier.</p> <p>Random distribution tables that use the random by generation method have an internally generated column called the RANDOM_DISTRIBUTION_KEY. This modifier does not apply to that column, only to other generated columns in the table. Values for the RANDOM_DISTRIBUTION_KEY will be regenerated unless explicitly referenced in the column list.</p>
generatedmissing	<p>If this modifier is specified, the utility assumes that the input data file contains no data for the generated columns (not even NULLs), and will therefore generate a value for each row. This modifier cannot be used with the generatedignore modifier.</p> <p>Random distribution tables that use the random by generation method have an internally generated column called the RANDOM_DISTRIBUTION_KEY. This modifier does not apply to that column, only to other generated columns in the table. Values for the RANDOM_DISTRIBUTION_KEY will be regenerated unless explicitly referenced in the column list.</p>
identityignore	<p>This modifier informs the import utility that data for the identity column is present in the data file but should be ignored. This results in all identity values being generated by the utility. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. This means that for GENERATED ALWAYS columns, no rows will be rejected. This modifier cannot be used with the identitymissing modifier.</p>
identitymissing	<p>If this modifier is specified, the utility assumes that the input data file contains no data for the identity column (not even NULLs), and will therefore generate a value for each row. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. This modifier cannot be used with the identityignore modifier.</p>

Table 6. Valid file type modifiers for the import utility: All file formats (continued)

Modifier	Description
implicitlyhiddeninclude	<p>If this modifier is specified, the utility assumes that the input data file contains data for the implicitly hidden columns and this data will also be imported. This modifier cannot be used with the implicitlyhiddenmissing modifier. See the Note: section for information about the precedence when multiple modifiers are specified.</p> <p>This modifier does not apply to the hidden RANDOM_DISTRIBUTION_KEY column of a random distribution tables that uses the random by generation method. The value for that column will be read from the input data file only if the column was explicitly referenced in the column list.</p>
implicitlyhiddenmissing	<p>If this modifier is specified, the utility assumes that the input data file does not contain data for the implicitly hidden columns and the utility will generate values for those hidden columns. This modifier cannot be used with the implicitlyhiddeninclude modifier. See the Note: section for information about the precedence when multiple modifiers are specified.</p> <p>This modifier does not apply to the hidden RANDOM_DISTRIBUTION_KEY column of a random distribution tables that uses the random by generation method. The value for that column will be read from the input data file only if the column was explicitly referenced in the column list.</p>
lobsinfile	<p><i>lob-path</i> specifies the path to the files containing LOB data.</p> <p>Each path contains at least one file that contains at least one LOB pointed to by a Lob Location Specifier (LLS) in the data file. The LLS is a string representation of the location of a LOB in a file stored in the LOB file path. The format of an LLS is <i>filename.ext.nnn.mmm/</i>, where <i>filename.ext</i> is the name of the file that contains the LOB, <i>nnn</i> is the offset in bytes of the LOB within the file, and <i>mmm</i> is the length of the LOB in bytes. For example, if the string db2exp.001.123.456/ is stored in the data file, the LOB is located at offset 123 in the file db2exp.001, and is 456 bytes long.</p> <p>The LOBS FROM clause specifies where the LOB files are located when the “lobsinfile” modifier is used. The LOBS FROM clause will implicitly activate the LOBSINFILE behavior. The LOBS FROM clause conveys to the IMPORT utility the list of paths to search for the LOB files while importing the data.</p> <p>To indicate a null LOB, enter the size as -1. If the size is specified as 0, it is treated as a 0 length LOB. For null LOBS with length of -1, the offset and the file name are ignored. For example, the LLS of a null LOB might be db2exp.001.7.-1/.</p>
no_type_id	Valid only when importing into a single sub-table. Typical usage is to export data from a regular table, and then to invoke an import operation (using this modifier) to convert the data into a single sub-table.
nodefaults	<p>If a source column for a target table column is not explicitly specified, and the table column is not nullable, default values are not loaded. Without this option, if a source column for one of the target table columns is not explicitly specified, one of the following occurs:</p> <ul style="list-style-type: none"> • If a default value can be specified for a column, the default value is loaded • If the column is nullable, and a default value cannot be specified for that column, a NULL is loaded • If the column is not nullable, and a default value cannot be specified, an error is returned, and the utility stops processing.
norowwarnings	Suppresses all warnings about rejected rows.

Table 6. Valid file type modifiers for the import utility: All file formats (continued)

Modifier	Description
periodignore	This modifier informs the import utility that data for the period columns is present in the data file but should be ignored. When this modifier is specified, all period column values are generated by the utility. This modifier cannot be used with the periodmissing modifier.
periodmissing	If this modifier is specified, the utility assumes that the input data file contains no data for the period columns. When this modifier is specified, all period column values are generated by the utility. This modifier cannot be used with the periodignore modifier.
rowchangetimestampignore	This modifier informs the import utility that data for the row change timestamp column is present in the data file but should be ignored. This results in all ROW CHANGE TIMESTAMP being generated by the utility. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT columns. This means that for GENERATED ALWAYS columns, no rows will be rejected. This modifier cannot be used with the rowchangetimestampmissing modifier.
rowchangetimestampmissing	If this modifier is specified, the utility assumes that the input data file contains no data for the row change timestamp column (not even NULLs), and will therefore generate a value for each row. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT columns. This modifier cannot be used with the rowchangetimestampignore modifier.
seclabelchar	Indicates that security labels in the input source file are in the string format for security label values rather than in the default encoded numeric format. IMPORT converts each security label into the internal format as it is loaded. If a string is not in the proper format the row is not loaded and a warning (SQLSTATE 01H53) is returned. If the string does not represent a valid security label that is part of the security policy protecting the table then the row is not loaded and a warning (SQLSTATE 01H53, SQLCODE SQL3243W)) is returned. This modifier cannot be specified if the seclabelname modifier is specified, otherwise the import fails and an error (SQLCODE SQL3525N) is returned.
seclabelname	Indicates that security labels in the input source file are indicated by their name rather than the default encoded numeric format. IMPORT will convert the name to the appropriate security label if it exists. If no security label exists with the indicated name for the security policy protecting the table the row is not loaded and a warning (SQLSTATE 01H53, SQLCODE SQL3244W) is returned. This modifier cannot be specified if the seclabelchar modifier is specified, otherwise the import fails and an error (SQLCODE SQL3525N) is returned. Note: If the file type is ASC, any spaces following the name of the security label will be interpreted as being part of the name. To avoid this use the striptblanks file type modifier to make sure the spaces are removed.
transactionidignore	This modifier informs the import utility that data for the TRANSACTION START ID column is present in the data file but should be ignored. When this modifier is specified, the value for the TRANSACTION START ID column is generated by the utility. This modifier cannot be used with the transactionidmissing modifier.
transactionidmissing	If this modifier is specified, the utility assumes that the input data file contains no data for the TRANSACTION START ID columns. When this modifier is specified, the value for the TRANSACTION START ID column is generated by the utility. This modifier cannot be used with the transactionidignore modifier.

Table 6. Valid file type modifiers for the import utility: All file formats (continued)

Modifier	Description
usedefaults	<p>If a source column for a target table column has been specified, but it contains no data for one or more row instances, default values are loaded. Examples of missing data are:</p> <ul style="list-style-type: none"> For DEL files: two adjacent column delimiters (",,") or two adjacent column delimiters separated by an arbitrary number of spaces (" , ") are specified for a column value. For DEL/ASC files: A row that does not have enough columns, or is not long enough for the original specification. <p>Note: For ASC files, NULL column values are not considered explicitly missing, and a default will not be substituted for NULL column values. NULL column values are represented by all space characters for numeric, date, time, and /timestamp columns, or by using the NULL INDICATOR for a column of any type to indicate the column is NULL.</p> <p>Without this option, if a source column contains no data for a row instance, one of the following occurs:</p> <ul style="list-style-type: none"> For DEL/ASC files: If the column is nullable, a NULL is loaded. If the column is not nullable, the utility rejects the row. <p>You cannot use the usedefaults file type modifier in INSERT_UPDATE mode if the input data is missing the value for a column that is part of a primary key when trying to update an existing row. The existing row is not updated and SQL3116W is returned.</p>

Table 7. Valid file type modifiers for the import utility: ASCII file formats (ASC/DEL)

Modifier	Description
codepage= <i>x</i>	<p><i>x</i> is an ASCII character string. The value is interpreted as the code page of the data in the input data set. Converts character data from this code page to the application code page during the import operation.</p> <p>The following rules apply:</p> <ul style="list-style-type: none"> For pure DBCS (graphic) mixed DBCS, and EUC, delimiters are restricted to the range of x00 to x3F, inclusive. nullindchar must specify symbols included in the standard ASCII set between code points x20 and x7F, inclusive. This refers to ASCII symbols and code points. <p>Note:</p> <ol style="list-style-type: none"> If data expansion occurs when the code page is converted from the application code page to the database code page, the data might be truncated and loss of data can occur.

Table 7. Valid file type modifiers for the import utility: ASCII file formats (ASC/DEL) (continued)

Modifier	Description
dateformat="x"	<p>x is the format of the date in the source file.² Valid date elements are:</p> <p>YYYY Year (four digits ranging from 0000 - 9999)</p> <p>M Month (one or two digits ranging from 1 - 12)</p> <p>MM Month (two digits ranging from 01 - 12; mutually exclusive with M)</p> <p>D Day (one or two digits ranging from 1 - 31)</p> <p>DD Day (two digits ranging from 01 - 31; mutually exclusive with D)</p> <p>DDD Day of the year (three digits ranging from 001 - 366; mutually exclusive with other day or month elements)</p> <p>A default value of 1 is assigned for each element that is not specified. Some examples of date formats are:</p> <p>"D-M-YYYY"</p> <p>"MM.DD.YYYY"</p> <p>"YYYYDDD"</p>
implieddecimal	<p>The location of an implied decimal point is determined by the column definition; it is no longer assumed to be at the end of the value. For example, the value 12345 is loaded into a DECIMAL(8,2) column as 123.45, <i>not</i> 12345.00.</p>
timeformat="x"	<p>x is the format of the time in the source file.² Valid time elements are:</p> <p>H Hour (one or two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system)</p> <p>HH Hour (two digits ranging from 00 - 12 for a 12 hour system, and 00 - 24 for a 24 hour system; mutually exclusive with H)</p> <p>M Minute (one or two digits ranging from 0 - 59)</p> <p>MM Minute (two digits ranging from 00 - 59; mutually exclusive with M)</p> <p>S Second (one or two digits ranging from 0 - 59)</p> <p>SS Second (two digits ranging from 00 - 59; mutually exclusive with S)</p> <p>SSSSS Second of the day after midnight (5 digits ranging from 00000 - 86400; mutually exclusive with other time elements)</p> <p>TT Meridian indicator (AM or PM)</p> <p>A default value of 0 is assigned for each element that is not specified. Some examples of time formats are:</p> <p>"HH:MM:SS"</p> <p>"HH.MM TT"</p> <p>"SSSSS"</p>

Table 7. Valid file type modifiers for the import utility: ASCII file formats (ASC/DEL) (continued)

Modifier	Description
timestampformat="x"	<p>x is the format of the time stamp in the source file.² Valid time stamp elements are:</p> <p>YYYY Year (four digits ranging from 0000 - 9999)</p> <p>M Month (one or two digits ranging from 1 - 12)</p> <p>MM Month (two digits ranging from 01 - 12; mutually exclusive with M and MMM)</p> <p>MMM Month (three-letter case-insensitive abbreviation for the month name; mutually exclusive with M and MM)</p> <p>D Day (one or two digits ranging from 1 - 31)</p> <p>DD Day (two digits ranging from 01 - 31; mutually exclusive with D)</p> <p>DDD Day of the year (three digits ranging from 001 - 366; mutually exclusive with other day or month elements)</p> <p>H Hour (one or two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system)</p> <p>HH Hour (two digits ranging from 00 - 12 for a 12 hour system, and 00 - 24 for a 24 hour system; mutually exclusive with H)</p> <p>M Minute (one or two digits ranging from 0 - 59)</p> <p>MM Minute (two digits ranging from 00 - 59; mutually exclusive with M, minute)</p> <p>S Second (one or two digits ranging from 0 - 59)</p> <p>SS Second (two digits ranging from 00 - 59; mutually exclusive with S)</p> <p>SSSSS Second of the day after midnight (5 digits ranging from 00000 - 86400; mutually exclusive with other time elements)</p> <p>U (1 to 12 times) Fractional seconds(number of occurrences of U represent the number of digits with each digit ranging from 0 to 9)</p> <p>TT Meridian indicator (AM or PM)</p> <p>A default value of 1 is assigned for unspecified YYYY, M, MM, D, DD, or DDD elements. A default value of 'Jan' is assigned to an unspecified MMM element. A default value of 0 is assigned for all other unspecified elements. Following is an example of a time stamp format:</p> <p>"YYYY/MM/DD HH:MM:SS.UUUUUU"</p> <p>The valid values for the MMM element include: 'jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov' and 'dec'. These values are case insensitive.</p> <p>The following example illustrates how to import data containing user defined date and time formats into a table called schedule:</p> <pre>db2 import from delfile2 of del modified by timestampformat="yyyy.mm.dd hh:mm tt" insert into schedule</pre>

Table 7. Valid file type modifiers for the import utility: ASCII file formats (ASC/DEL) (continued)

Modifier	Description
usegraphiccodepage	<p>If usegraphiccodepage is given, the assumption is made that data being imported into graphic or double-byte character large object (DBCLOB) data fields is in the graphic code page. The rest of the data is assumed to be in the character code page. The graphic code page is associated with the character code page. IMPORT determines the character code page through either the codepage modifier, if it is specified, or through the code page of the application if the codepage modifier is not specified.</p> <p>This modifier should be used in conjunction with the delimited data file generated by drop table recovery only if the table being recovered has graphic data.</p> <p>Restrictions</p> <p>The usegraphiccodepage modifier MUST NOT be specified with DEL files created by the EXPORT utility, as these files contain data encoded in only one code page. The usegraphiccodepage modifier is also ignored by the double-byte character large objects (DBCLOBs) in files.</p>
xmlchar	<p>Specifies that XML documents are encoded in the character code page.</p> <p>This option is useful for processing XML documents that are encoded in the specified character code page but do not contain an encoding declaration.</p> <p>For each document, if a declaration tag exists and contains an encoding attribute, the encoding must match the character code page, otherwise the row containing the document will be rejected. Note that the character code page is the value specified by the codepage file type modifier, or the application code page if it is not specified. By default, either the documents are encoded in Unicode, or they contain a declaration tag with an encoding attribute.</p>
xmlgraphic	<p>Specifies that XML documents are encoded in the specified graphic code page.</p> <p>This option is useful for processing XML documents that are encoded in a specific graphic code page but do not contain an encoding declaration.</p> <p>For each document, if a declaration tag exists and contains an encoding attribute, the encoding must match the graphic code page, otherwise the row containing the document will be rejected. Note that the graphic code page is the graphic component of the value specified by the codepage file type modifier, or the graphic component of the application code page if it is not specified. By default, documents are either encoded in Unicode, or they contain a declaration tag with an encoding attribute.</p> <p>Note: If the xmlgraphic modifier is specified with the IMPORT command, the XML document to be imported must be encoded in the UTF-16 code page. Otherwise, the XML document may be rejected with a parsing error, or it may be imported into the table with data corruption.</p>

Table 8. Valid file type modifiers for the import utility: ASC (non-delimited ASCII) file format

Modifier	Description
nochecklengths	<p>If nochecklengths is specified, an attempt is made to import each row, even if the source data has a column definition that exceeds the size of the target table column. Such rows can be successfully imported if code page conversion causes the source data to shrink; for example, 4-byte EUC data in the source could shrink to 2-byte DBCS data in the target, and require half the space. This option is particularly useful if it is known that the source data will fit in all cases despite mismatched column definitions.</p>

Table 8. Valid file type modifiers for the import utility: ASC (non-delimited ASCII) file format (continued)

Modifier	Description
nullindchar= <i>x</i>	<p><i>x</i> is a single character. Changes the character denoting a null value to <i>x</i>. The default value of <i>x</i> is Y.³</p> <p>This modifier is case sensitive for EBCDIC data files, except when the character is an English letter. For example, if the null indicator character is specified to be the letter N, then n is also recognized as a null indicator.</p>
reclen= <i>x</i>	<p><i>x</i> is an integer with a maximum value of 32 767. <i>x</i> characters are read for each row, and a new-line character is not used to indicate the end of the row.</p>
striptblanks	<p>Truncates any trailing blank spaces when loading data into a variable-length field. If this option is not specified, blank spaces are kept.</p> <p>In the following example, striptblanks causes the import utility to truncate trailing blank spaces:</p> <pre>db2 import from myfile.asc of asc modified by striptblanks method 1 (1 10, 12 15) messages msgs.txt insert into staff</pre> <p>This option cannot be specified together with striptnulls. These are mutually exclusive options. This option replaces the obsolete t option, which is supported for earlier compatibility only.</p>
striptnulls	<p>Truncates any trailing NULLs (0x00 characters) when loading data into a variable-length field. If this option is not specified, NULLs are kept.</p> <p>This option cannot be specified together with striptblanks. These are mutually exclusive options. This option replaces the obsolete padwithzero option, which is supported for earlier compatibility only.</p>

Table 9. Valid file type modifiers for the import utility: DEL (delimited ASCII) file format

Modifier	Description
chardel <i>x</i>	<p><i>x</i> is a single character string delimiter. The default value is a double quotation mark ("). The specified character is used in place of double quotation marks to enclose a character string.³⁴ If you want to explicitly specify the double quotation mark as the character string delimiter, it should be specified as follows:</p> <pre>modified by chardel'"</pre> <p>The single quotation mark (') can also be specified as a character string delimiter. In the following example, chardel'' causes the import utility to interpret any single quotation mark (') it encounters as a character string delimiter:</p> <pre>db2 "import from myfile.del of del modified by chardel'' method p (1, 4) insert into staff (id, years)"</pre>
coldel <i>x</i>	<p><i>x</i> is a single character column delimiter. The default value is a comma (.). The specified character is used in place of a comma to signal the end of a column.³⁴</p> <p>In the following example, coldel; causes the import utility to interpret any semicolon (;) it encounters as a column delimiter:</p> <pre>db2 import from myfile.del of del modified by coldel; messages msgs.txt insert into staff</pre>
decplusblank	<p>Plus sign character. Causes positive decimal values to be prefixed with a blank space instead of a plus sign (+). The default action is to prefix positive decimal values with a plus sign.</p>

Table 9. Valid file type modifiers for the import utility: DEL (delimited ASCII) file format (continued)

Modifier	Description
decptx	<p><i>x</i> is a single character substitute for the period as a decimal point character. The default value is a period (.). The specified character is used in place of a period as a decimal point character.³⁴</p> <p>In the following example, decpt; causes the import utility to interpret any semicolon (;) it encounters as a decimal point:</p> <pre>db2 "import from myfile.del of del modified by chardel'" decpt; messages msgs.txt insert into staff"</pre>
delprioritychar	<p>The current default priority for delimiters is: record delimiter, character delimiter, column delimiter. This modifier protects existing applications that depend on the older priority by reverting the delimiter priorities to: character delimiter, record delimiter, column delimiter. Syntax:</p> <pre>db2 import ... modified by delprioritychar ...</pre> <p>For example, given the following DEL data file:</p> <pre>"Smith, Joshua",4000,34.98<row delimiter> "Vincent,<row delimiter>, is a manager", 4005,44.37<row delimiter></pre> <p>With the delprioritychar modifier specified, there will be only two rows in this data file. The second <row delimiter> will be interpreted as part of the first data column of the second row, while the first and the third <row delimiter> are interpreted as actual record delimiters. If this modifier is <i>not</i> specified, there will be three rows in this data file, each delimited by a <row delimiter>.</p>
keepblanks	<p>Preserves the leading and trailing blanks in each field of type CHAR, VARCHAR, LONG VARCHAR, or CLOB. Without this option, all leading and trailing blanks that are not inside character delimiters are removed, and a NULL is inserted into the table for all blank fields.</p>
nochardel	<p>The import utility will assume all bytes found between the column delimiters to be part of the column's data. Character delimiters will be parsed as part of column data. This option should not be specified if the data was exported using Db2 (unless nochardel was specified at export time). It is provided to support vendor data files that do not have character delimiters. Improper usage might result in data loss or corruption.</p> <p>This option cannot be specified with chardelx, delprioritychar or nodoubledel. These are mutually exclusive options.</p>
nodoubledel	<p>Suppresses recognition of double character delimiters.</p>

Table 10. Valid file type modifiers for the import utility: IXF file format

Modifier	Description
forcein	<p>Directs the utility to accept data despite code page mismatches, and to suppress translation between code pages.</p> <p>Fixed length target fields are checked to verify that they are large enough for the data. If nochecklengths is specified, no checking is done, and an attempt is made to import each row.</p>
indexixf	<p>Directs the utility to drop all indexes currently defined on the existing table, and to create new ones from the index definitions in the PC/IXF file. This option can only be used when the contents of a table are being replaced. It cannot be used with a view, or when a <i>insert-column</i> is specified.</p>

Table 10. Valid file type modifiers for the import utility: IXF file format (continued)

Modifier	Description
indexschema= <i>schema</i>	Uses the specified <i>schema</i> for the index name during index creation. If <i>schema</i> is not specified (but the keyword <i>indexschema</i> is specified), uses the connection user ID. If the keyword is not specified, uses the schema in the IXF file.
nochecklengths	If <i>nochecklengths</i> is specified, an attempt is made to import each row, even if the source data has a column definition that exceeds the size of the target table column. Such rows can be successfully imported if code page conversion causes the source data to shrink; for example, 4-byte EUC data in the source could shrink to 2-byte DBCS data in the target, and require half the space. This option is particularly useful if it is known that the source data will fit in all cases despite mismatched column definitions.
forcecreate	Specifies that the table should be created with possible missing or limited information after returning SQL3311N during an import operation.

Table 11. IMPORT behavior when using codepage and usegraphiccodepage

codepage=N	usegraphiccodepage	IMPORT behavior
Absent	Absent	All data in the file is assumed to be in the application code page.
Present	Absent	All data in the file is assumed to be in code page N. Warning: Graphic data will be corrupted when imported into the database if N is a single-byte code page.
Absent	Present	Character data in the file is assumed to be in the application code page. Graphic data is assumed to be in the code page of the application graphic data. If the application code page is single-byte, then all data is assumed to be in the application code page. Warning: If the application code page is single-byte, graphic data will be corrupted when imported into the database, even if the database contains graphic columns.
Present	Present	Character data is assumed to be in code page N. Graphic data is assumed to be in the graphic code page of N. If N is a single-byte or double-byte code page, then all data is assumed to be in code page N. Warning: Graphic data will be corrupted when imported into the database if N is a single-byte code page.

Note:

1. The import utility does not issue a warning if an attempt is made to use unsupported file types with the **MODIFIED BY** option. If this is attempted, the import operation fails, and an error code is returned.
2. Double quotation marks around the date format string are mandatory. Field separators cannot contain any of the following: a-z, A-Z, and 0-9. The field separator should not be the same as the character delimiter or field delimiter in the DEL file format. A field separator is optional if the start and end

positions of an element are unambiguous. Ambiguity can exist if (depending on the modifier) elements such as D, H, M, or S are used, because of the variable length of the entries.

For time stamp formats, care must be taken to avoid ambiguity between the month and the minute descriptors, since they both use the letter M. A month field must be adjacent to other date fields. A minute field must be adjacent to other time fields. Following are some ambiguous time stamp formats:

```
"M" (could be a month, or a minute)
"M:M" (Which is which?)
"M:YYYY:M" (Both are interpreted as month.)
"S:M:YYYY" (adjacent to both a time value and a date value)
```

In ambiguous cases, the utility will report an error message, and the operation will fail.

Following are some unambiguous time stamp formats:

```
"M:YYYY" (Month)
"S:M" (Minute)
"M:YYYY:S:M" (Month....Minute)
"M:H:YYYY:M:D" (Minute....Month)
```

Some characters, such as double quotation marks and back slashes, must be preceded by an escape character (for example, \).

3. Character values provided for the `chardel`, `codel`, or `decpt` file type modifiers must be specified in the code page of the source data.

The character code point (instead of the character symbol), can be specified using the syntax `xJJ` or `0xJJ`, where JJ is the hexadecimal representation of the code point. For example, to specify the # character as a column delimiter, use one of the following statements:

```
... modified by codel# ...
... modified by codel0x23 ...
... modified by codelX23 ...
```

4. *Delimiter considerations for moving data* lists restrictions that apply to the characters that can be used as delimiter overrides.
5. The following file type modifiers are not allowed when importing into a nickname:
 - `indexixf`
 - `indexschema`
 - `dldelfiletype`
 - `nodefaults`
 - `usedefaults`
 - `no_type_idfiletype`
 - `generatedignore`
 - `generatedmissing`
 - `identityignore`
 - `identitymissing`
 - `lobsinfile`
6. The **CREATE** mode is not supported for XML columns.
7. All XML data must reside in XML files that are separate from the main data file. An XML Data Specifier (XDS) (or a NULL value) must exist for each XML column in the main data file.
8. XML documents are assumed to be in Unicode format or to contain a declaration tag that includes an encoding attribute, unless the `XMLCHAR` or `XMLGRAPHIC` file type modifier is specified.

9. Rows containing documents that are not well-formed will be rejected.
10. If the **XMLVALIDATE** option is specified, documents that successfully validate against their matching schema will be annotated with the schema information as they are inserted. Rows containing documents that fail to validate against their matching schema will be rejected. To successfully perform the validation, the privileges held by the user invoking the import must include at least one of the following:
 - DBADM authority
 - USAGE privilege on the XML schema to be used in the validation
11. When multiple modifiers suffixed with **ignore**, **include**, **missing**, and **override** are specified, they are applied in the order that they are listed. In the following statement, data for implicitly hidden columns that are not identity columns is included in the input data. While data for all identity columns, regardless of their implicitly hidden status, is not.

```
db2 import from delfile1 of del modified by
    implicitlyhiddeninclude identitymissing insert into table1
```

However, changing the order of the file type modifiers in the following statement means that data for all implicitly hidden columns (including hidden identity columns) is included in the input data. While data for identity columns that are not implicitly hidden is not.

```
db2 import from delfile1 of del modified by
    identitymissing implicitlyhiddeninclude insert into table1
```

If the DB2_DMU_DEFAULT registry variable is set to **IMPLICITLYHIDDENINCLUDE**, then:

```
db2set DB2_DMU_DEFAULT=IMPLICITLYHIDDENINCLUDE
db2 import from delfile1 of del modified by identitymissing insert into table1
```

is equivalent to:

```
db2 import from delfile1 of del modified by
    implicitlyhiddeninclude identitymissing insert into table1
```

INITIALIZE TAPE command using the ADMIN_CMD procedure:

Initializes tapes for backup and restore operations to streaming tape devices. This command is only supported on Windows operating systems.

Authorization

One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMAINT

Required connection

None

Command syntax

```
►►—INITIALIZE TAPE—┐ON—device┘┐USING—blksize┘┐
```

Command parameters

ON device

Specifies a valid tape device name. The default value is `\\.\TAPE0`.

USING blksize

Specifies the block size for the device, in bytes. The device is initialized to use the block size specified, if the value is within the supported range of block sizes for the device.

The buffer size specified for the **BACKUP DATABASE** command and for **RESTORE DATABASE** must be divisible by the block size specified here.

If a value for this parameter is not specified, the device is initialized to use its default block size. If a value of zero is specified, the device is initialized to use a variable length block size; if the device does not support variable length block mode, an error is returned.

When backing up to tape, use of a variable block size is currently not supported. If you must use this option, ensure that you have well tested procedures in place that enable you to recover successfully, using backup images that were created with a variable block size.

When using a variable block size, you must specify a backup buffer size that is less than or equal to the maximum limit for the tape devices that you are using. For optimal performance, the buffer size must be equal to the maximum block size limit of the device being used.

LOAD command using the ADMIN_CMD procedure:

Efficiently loads large amounts of data into a Db2 table.

The **LOAD** command loads data at the page level, bypasses trigger firing and logging, and delays constraint checking and index building until after the data is loaded into the Db2 table.

Data stored on the server can be in the form of a file, tape, or named pipe. Data stored on a remotely connected client can be in the form of a fully qualified file or named pipe. Data can also be loaded from a user-defined cursor or by using a user-written script or application. If the **COMPRESS** attribute for the table is set to **YES**, the data loaded is subject to compression on every data and database partition for which a dictionary exists in the table, including data in the XML storage object of the table.

Quick link to “File type modifiers for the load utility” on page 108.

Restrictions

The load utility does not support loading data at the hierarchy level. The load utility is not compatible with range-clustered tables. The load utility does not support the **NOT LOGGED INITIALLY** parameter for the **CREATE TABLE** or **ALTER TABLE** statements.

Scope

This command can be issued against multiple database partitions in a single request.

Authorization

One or more of the following authorities:

- DATAACCESS
- LOAD authority on the database and the following privileges:
 - INSERT privilege on the table when the load utility is invoked in INSERT mode, TERMINATE mode (to terminate a previous load insert operation), or RESTART mode (to restart a previous load insert operation)
 - INSERT and DELETE privilege on the table when the load utility is invoked in REPLACE mode, TERMINATE mode (to terminate a previous load replace operation), or RESTART mode (to restart a previous load replace operation)
 - INSERT privilege on the exception table, if such a table is used as part of the load operation.
- To load data into a table that has protected columns, the session authorization ID must have LBAC credentials directly or indirectly through a group or a role that allow write access to all protected columns in the table. Otherwise the load fails and an error (SQLSTATE 5U014) is returned.
- To load data into a table that has protected rows, the session authorization ID must hold a security label that meets these criteria:
 - The security label is part of the security policy protecting the table.
 - The security label was granted to the session authorization ID directly or indirectly through a group or a role for write access or for all access.

If the session authorization ID does not hold such a security label, then the load fails and an error (SQLSTATE 5U014) is returned. The security label protects a loaded row if the session authorization ID LBAC credentials do not allow it to write to the security label that protects that row in the data. This does not happen, however, when the security policy protecting the table was created with the RESTRICT NOT AUTHORIZED WRITE SECURITY LABEL option of the CREATE SECURITY POLICY statement. In this case the load fails and an error (SQLSTATE 42519) is returned.

When you load data into a table with protected rows, the target table has one column with a data type of DB2SECURITYLABEL. If the input row of data does not contain a value for that column, that row is rejected unless the usedefaults file type modifier is specified in the load command, in which case the security label you hold for write access from the security policy protecting the table is used. If you do not hold a security label for write access, the row is rejected and processing continues on to the next row

- For a table with protected rows, if the REPLACE option is specified, the session authorization ID must have the authority to drop the table.
- If the LOCK WITH FORCE option is specified, any of SYSMAINT, SYSCTRL, or SYSADM authority is required.
- If the table has row access control activated, then **LOAD REPLACE** on that table would require the ability to drop the table. Specifically, you must have either CONTROL or DBADM on the table.

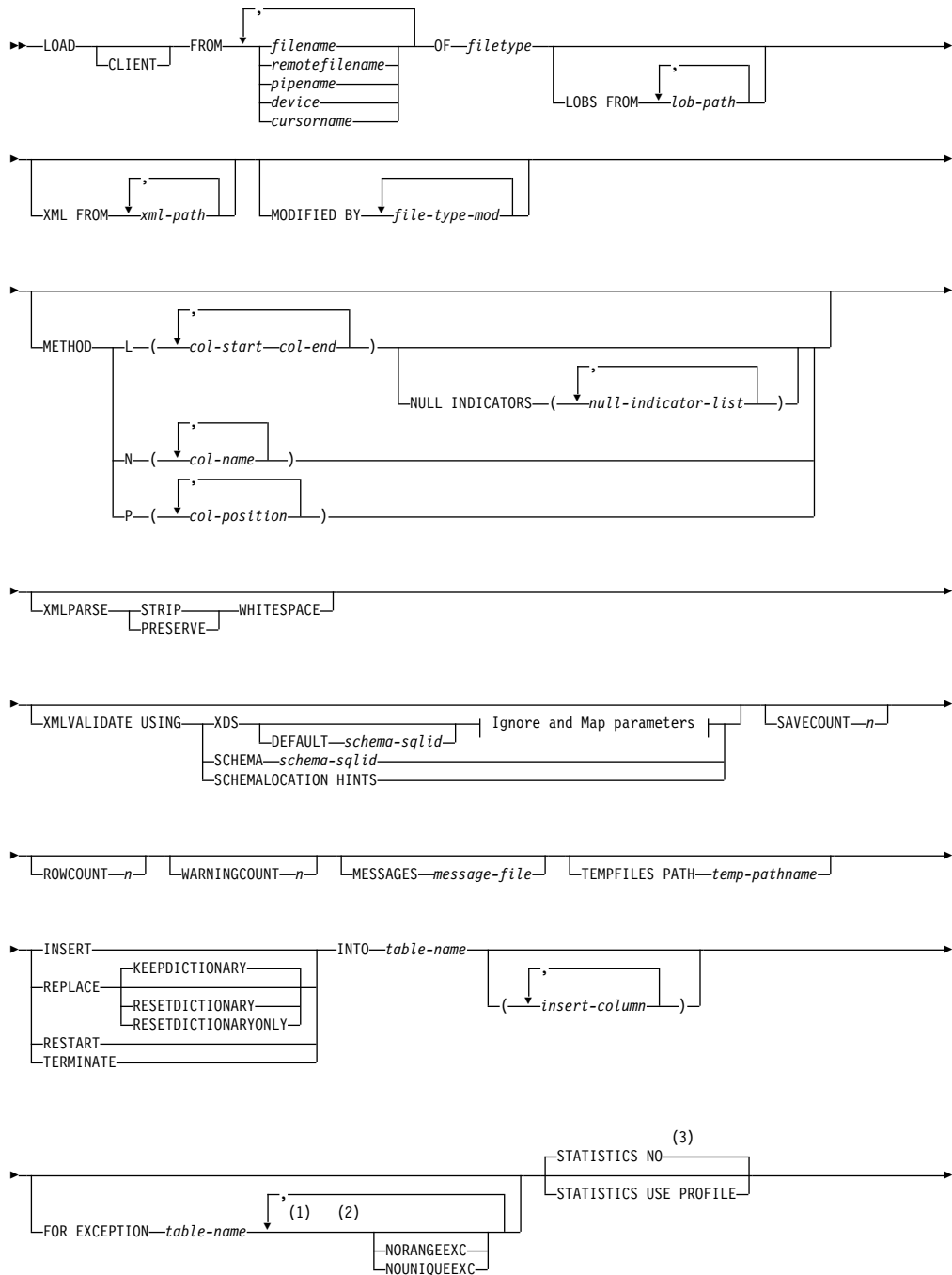
Since all load processes (and all Db2 server processes, in general) are owned by the instance owner, and all of these processes use the identification of the instance owner to access needed files, the instance owner must have read access to input data files. These input data files must be readable by the instance owner, regardless of who invokes the command.

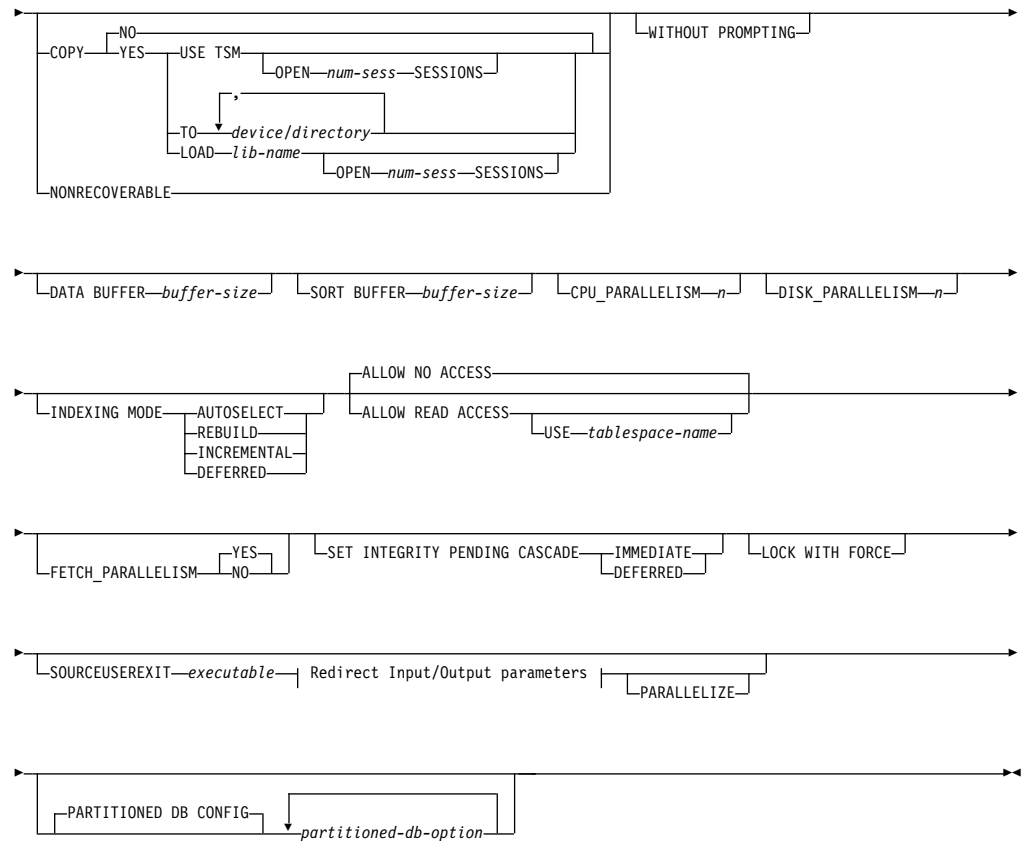
Required connection

Database. If implicit connect is enabled, a connection to the default database is established.

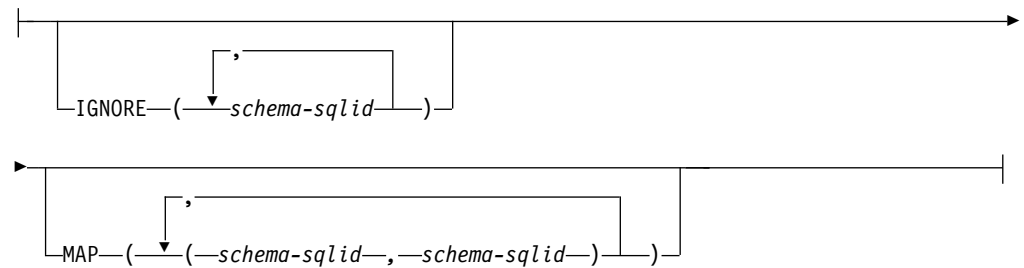
Instance. An explicit attachment is not required. If a connection to the database has been established, an implicit attachment to the local instance is attempted.

Command syntax

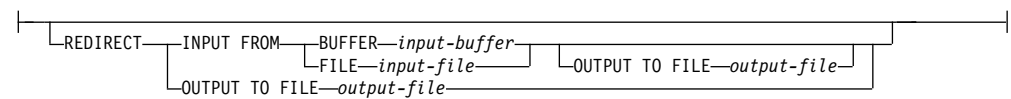




Ignore and Map parameters:



Redirect Input/Output parameters:



Notes:

- 1 These keywords can appear in any order.
- 2 Each of these keywords can appear only once.
- 3 For column-organized tables, the default is the **STATISTICS USE PROFILE** parameter.

Command parameters

CLIENT Specifies that the data to be loaded resides on a remotely connected client. This option is ignored if the load operation is not being invoked from a remote client. This option is ignored if specified in conjunction with the **CURSOR** file type.

Note:

1. The **dumpfile** and **lobsinfile** modifiers and the **XML FROM** option refer to files on the server even when the **CLIENT** keyword is specified.
2. Code page conversion is not performed during a remote load operation. If the code page of the data is different from that of the server, the data code page should be specified using the **codepage** modifier.

In the following example, a data file (/u/user/data.del) residing on a remotely connected client is to be loaded into MYTABLE on the server database:

```
db2 load client from /u/user/data.del of del
modified by codepage=850 insert into mytable
```

FROM *filename* | *remotefilename* | *pipename* | *device* | *cursorname*

Specifies the file, pipe, device, or cursor referring to an SQL statement that contains the data being loaded. If the input source is a file, pipe, or device, it must reside on the database partition where the database resides, unless the **CLIENT** option is specified.

A *remotefilename* refers to a file that is on remote storage, such as IBM SoftLayer Object Storage or Amazon Simple Storage Service (S3), and is being accessed using a storage access alias. Local staging space is required to temporarily store the file that is transferred from the remote storage server; refer to Remote storage requirements. The syntax of remote file names is:

```
DB2REMOTE://<alias>//<storage-path>/<file-name>
```

If several names are specified, they will be processed in sequence. If the last item specified is a tape device, the user is prompted for another tape. Valid response options are:

- c** Continue. Continue using the device that generated the warning message (for example, when a new tape has been mounted).
- d** Device terminate. Stop using the device that generated the warning message (for example, when there are no more tapes).
- t** Terminate. Terminate all devices.

Note:

- Use the fully qualified file name. If the server is remote, the fully qualified file name must be used. If the database resides on the same database partition as the caller, relative paths can be used.
- If data is exported into a file using the **EXPORT** command using the **ADMIN_CMD** procedure, the data file is owned by the fenced user ID. This file is not usually accessible by the instance owner. To run the **LOAD** from CLP or the **ADMIN_CMD** procedure, the data file must be accessible by the instance owner ID, so read access to the data file must be granted to the instance owner.
- Loading data from multiple IXF files is supported if the files are physically separate, but logically one file. It is *not* supported if the files

are both logically and physically separate. If more than one logically and physically separate files are specified, then any file after the first one is ignored. (Multiple physical files would be considered logically one if they were all created with one invocation of the **EXPORT** command.)

- If loading data that resides on a client machine, the data must be in the form of either a fully qualified file or a named pipe.
- When loading XML data from files into tables in a partitioned database environment, the XML data files must be read-accessible to all the database partitions where loading is taking place.

OF *filetype*

Specifies the format of the data:

- ASC (non-delimited ASCII format)
- DEL (delimited ASCII format)
- IXF (Integration Exchange Format, PC version) is a binary format that is used exclusively by Db2 databases.
- CURSOR (a cursor declared against a SELECT or VALUES statement).

Note:

- When using a CURSOR file type to load XML data into a table in a distributed database environment, the PARTITION_ONLY and LOAD_ONLY modes are not supported.
- When performing a load using the CURSOR file type where the DATABASE keyword was specified during the DECLARE CURSOR statement, load internally creates a separate application to fetch the data; whereas when the DATABASE keyword is not specified, load fetches data within the same application. This difference between the two cases can also cause locking behavior difference. In particular, if you currently specify the DATABASE keyword using the same database as the currently connected database (and same userid and password as the current connection, if specified), there might be cases where you get into a lock issue (such as a lock wait or lock timeout, depending on the database configuration) which can be worked around by omitting the DATABASE keyword.

LOBS FROM *lob-path*

The path to the data files containing LOB values to be loaded. The path must end with a slash. If the **CLIENT** option is specified, the path must be fully qualified. The names of the LOB data files are stored in the main data file (ASC, DEL, or IXF), in the column that will be loaded into the LOB column. The maximum number of paths that can be specified is 999. This will implicitly activate the **LOBSINFILE** behavior.

This option is ignored when specified in conjunction with the CURSOR file type.

MODIFIED BY *file-type-mod*

Specifies file type modifier options. See “File type modifiers for the load utility” on page 108.

METHOD

- L** Specifies the start and end column numbers from which to load data. A column number is a byte offset from the beginning of a row of data. It is numbered starting from 1. This method can only be used with ASC files, and is the only valid method for that file type.

NULL INDICATORS *null-indicator-list*

This option can only be used when the **METHOD L** parameter is specified; that is, the input file is an ASC file). The null indicator list is a comma-separated list of positive integers specifying the column number of each null indicator field. The column number is the byte offset of the null indicator field from the beginning of a row of data. There must be one entry in the null indicator list for each data field defined in the **METHOD L** parameter. A column number of zero indicates that the corresponding data field always contains data.

A value of Y in the NULL indicator column specifies that the column data is NULL. Any character *other than* Y in the NULL indicator column specifies that the column data is not NULL, and that column data specified by the **METHOD L** option will be loaded.

The NULL indicator character can be changed using the **MODIFIED BY** option.

- N** Specifies the names of the columns in the data file to be loaded. The case of these column names must match the case of the corresponding names in the system catalogs. Each table column that is not nullable should have a corresponding entry in the **METHOD N** list. For example, given data fields F1, F2, F3, F4, F5, and F6, and table columns C1 INT, C2 INT NOT NULL, C3 INT NOT NULL, and C4 INT, method N (F2, F1, F4, F3) is a valid request, while method N (F2, F1) is not valid. This method can only be used with file types IXF or CURSOR.
- P** Specifies the field numbers (numbered from 1) of the input data fields to be loaded. Each table column that is not nullable should have a corresponding entry in the **METHOD P** list. For example, given data fields F1, F2, F3, F4, F5, and F6, and table columns C1 INT, C2 INT NOT NULL, C3 INT NOT NULL, and C4 INT, method P (2, 1, 4, 3) is a valid request, while method P (2, 1) is not valid. This method can only be used with file types IXF, DEL, or CURSOR, and is the only valid method for the DEL file type.

For each of the fields specified by method P, you need to define a corresponding column in the action statement, unless all columns are accounted for or the first *x* columns are going to be loaded, as shown in the following example:

```
db2 import from datafile1.del of del method P(1, 3, 4)
  replace into table1 (c1, c3, c4)
```

XML FROM *xml-path*

Specifies one or more paths that contain the XML files. XDSs are contained in the main data file (ASC, DEL, or IXF), in the column that will be loaded into the XML column.

XMLPARSE

Specifies how XML documents are parsed. If this option is not specified, the parsing behavior for XML documents will be determined by the value of the CURRENT IMPLICIT XMLPARSE OPTION special register.

STRIP WHITESPACE

Specifies to remove whitespace when the XML document is parsed.

PRESERVE WHITESPACE

Specifies not to remove whitespace when the XML document is parsed.

XMLVALIDATE

Specifies that XML documents are validated against a schema, when applicable.

USING XDS

XML documents are validated against the XML schema identified by the XML Data Specifier (XDS) in the main data file. By default, if the **XMLVALIDATE** option is invoked with the **USING XDS** clause, the schema used to perform validation will be determined by the SCH attribute of the XDS. If an SCH attribute is not present in the XDS, no schema validation will occur unless a default schema is specified by the **DEFAULT** clause.

The **DEFAULT**, **IGNORE**, and **MAP** clauses can be used to modify the schema determination behavior. These three optional clauses apply directly to the specifications of the XDS, and not to each other. For example, if a schema is selected because it is specified by the **DEFAULT** clause, it will not be ignored if also specified by the **IGNORE** clause. Similarly, if a schema is selected because it is specified as the first part of a pair in the **MAP** clause, it will not be re-mapped if also specified in the second part of another **MAP** clause pair.

USING SCHEMA *schema-sqlid*

XML documents are validated against the XML schema with the specified SQL identifier. In this case, the SCH attribute of the XML Data Specifier (XDS) will be ignored for all XML columns.

USING SCHEMALOCATION HINTS

XML documents are validated against the schemas identified by XML schema location hints in the source XML documents. If a schemaLocation attribute is not found in the XML document, no validation will occur. When the **USING SCHEMALOCATION HINTS** clause is specified, the SCH attribute of the XML Data Specifier (XDS) will be ignored for all XML columns.

See examples of the **XMLVALIDATE** option in the following section.

IGNORE *schema-sqlid*

This option can only be used when the **USING XDS** parameter is specified. The **IGNORE** clause specifies a list of one or more schemas to ignore if they are identified by an SCH attribute. If an SCH attribute exists in the XML Data Specifier for a loaded XML document, and the schema identified by the SCH attribute is included in the list of schemas to ignore, then no schema validation will occur for the loaded XML document.

Note:

If a schema is specified in the **IGNORE** clause, it cannot also be present in the left side of a schema pair in the **MAP** clause.

The **IGNORE** clause applies only to the XDS. A schema that is mapped by the **MAP** clause will not be subsequently ignored if specified by the **IGNORE** clause.

DEFAULT *schema-sqlid*

This option can only be used when the **USING XDS** parameter is specified. The schema specified through the **DEFAULT** clause identifies a schema to use for validation when the XML Data Specifier (XDS) of a loaded XML document does not contain an SCH attribute identifying an XML Schema.

The **DEFAULT** clause takes precedence over the **IGNORE** and **MAP** clauses. If an XDS satisfies the **DEFAULT** clause, the **IGNORE** and **MAP** specifications will be ignored.

MAP *schema-sqlid*

This option can only be used when the **USING XDS** parameter is specified. Use the **MAP** clause to specify alternate schemas to use in place of those specified by the SCH attribute of an XML Data Specifier (XDS) for each loaded XML document. The **MAP** clause specifies a list of one or more schema pairs, where each pair represents a mapping of one schema to another. The first schema in the pair represents a schema that is referred to by an SCH attribute in an XDS. The second schema in the pair represents the schema that should be used to perform schema validation.

If a schema is present in the left side of a schema pair in the **MAP** clause, it cannot also be specified in the **IGNORE** clause.

Once a schema pair mapping is applied, the result is final. The mapping operation is non-transitive, and therefore the schema chosen will not be subsequently applied to another schema pair mapping.

A schema cannot be mapped more than once, meaning that it cannot appear on the left side of more than one pair.

SAVECOUNT *n*

Specifies that the load utility is to establish consistency points after every *n* rows. This value is converted to a page count, and rounded up to intervals of the extent size. Since a message is issued at each consistency point, this option should be selected if the load operation is monitored using **LOAD QUERY**. If the value of *n* is not sufficiently high, the synchronization of activities performed at each consistency point might impact performance.

The default value is zero, meaning that no consistency points are established, unless necessary.

This option is not allowed when specified in conjunction with the **CURSOR** file type or when loading a table containing an XML column.

The **SAVECOUNT** parameter is not supported for column-organized tables.

ROWCOUNT *n*

Specifies the number of *n* physical records in the file to be loaded. If the **anyorder** file type modifier is enabled any *n* rows can be loaded from the file. Otherwise the first *n* rows are loaded.

Note: The **anyorder** file type modifier is enabled by default for certain table types. For more information, see **anyorder**.

WARNINGCOUNT *n*

Stops the load operation after *n* warnings. Set this parameter if no warnings are expected, but verification that the correct file and table are being used is desired. If the load file or the target table is specified incorrectly, the load utility generates a warning for each row that it

attempts to load, which causes the load to fail. If *n* is zero, or this option is not specified, the load operation continues regardless of the number of warnings issued.

If the load operation is stopped because the threshold of warnings is encountered, another load operation can be started in **RESTART** mode. The load operation automatically continues from the last consistency point. Alternatively, another load operation can be initiated in **REPLACE** mode, starting at the beginning of the input file.

In a partitioned database environment, a **LOAD** operation can have multiple load and partition agents. Each agent has a **WARNINGCOUNT** value. If the value of *n* is reached on a single agent, the **LOAD** operation fails. The *n* values are not cumulative. For example, if *n* is 3 and there are two agents, each with a **WARNINGCOUNT** of 2, the **LOAD** operation is successful.

MESSAGES *message-file*

Specifies the destination for warning and error messages that occur during the load operation. If a message file is not specified, messages are written to standard output unless the instance is configured as a partitioned database environment. If the complete path to the file is not specified, the load utility uses the current directory and the default drive as the destination. If the name of a file that already exists is specified, the utility appends the information.

The message file is usually populated with messages at the end of the load operation and, as such, is not suitable for monitoring the progress of the operation.

TEMPFILES PATH *temp-pathname*

Specifies the name of the path to be used when creating temporary files during a load operation, and should be fully qualified according to the server database partition.

Temporary files take up file system space. Sometimes, this space requirement is quite substantial. The following list provides an estimate of how much file system space you should allocate for all temporary files:

- 136 bytes for each message that the load utility generates.
- 15 KB of overhead if the data file contains long field data or LOBs. This quantity can grow significantly if you specify the **INSERT** parameter and there is a large amount of long field or LOB data already in the table.
- On the server, storage space that is equivalent to the raw size of the input data if the column compression dictionary is being built and the data source, such as a pipe or a socket, cannot be reopened.

INSERT Adds the loaded data to the table without changing the existing table data.

A **LOAD INSERT** operation into a column-organized table maintains table statistics by default if the table was empty at the start of the load operation.

REPLACE

Deletes all data from the table, and inserts the new data. The table definition and index definitions are not changed. If you specify this parameter when moving data between hierarchies, you can replace only the data for an entire hierarchy, not individual subtables.

You cannot use this parameter to load data into system-period temporal tables.

A **LOAD REPLACE** operation into a column-organized table maintains table statistics by default.

KEEPDICTIONARY

An existing compression dictionary is preserved across the **LOAD REPLACE** operation.

This option is the default for row-organized tables.

Provided the table **COMPRESS** attribute is **YES**, the newly replaced data is subject to being compressed using the dictionary that existed before the invocation of the load. If no dictionary previously existed in the table, a new dictionary is built using the data that is being replaced into the table as long as the table **COMPRESS** attribute is **YES**.

The amount of data that is required to build the compression dictionary in this case is subject to the policies of ADC. This data is populated into the table as uncompressed. Once the dictionary is inserted into the table, the remaining data to be loaded is subject to being compressed with this dictionary. For a summary, see the following table.

Table 12. LOAD REPLACE KEEPDICTIONARY

Compress	Table row data dictionary exists	XML storage object dictionary exists¹	Compression dictionary	Data compression
YES	YES	YES	Preserve table row data and XML dictionaries.	Data to be loaded is subject to compression.
YES	YES	NO	Preserve table row data dictionary and build a new XML dictionary.	Table row data to be loaded is subject to compression. After XML dictionary is built, remaining XML data to be loaded is subject to compression.
YES	NO	YES	Build table row data dictionary and preserve XML dictionary.	After table row data dictionary is built, remaining table row data to be loaded is subject to compression. XML data to be loaded is subject to compression.
YES	NO	NO	Build new table row data and XML dictionaries.	After dictionaries are built, remaining data to be loaded is subject to compression.
NO	YES	YES	Preserve table row data and XML dictionaries.	Data to be loaded is not compressed.
NO	YES	NO	Preserve table row data dictionary.	Data to be loaded is not compressed.
NO	NO	YES	No effect on table row dictionary. Preserve XML dictionary.	Data to be loaded is not compressed.
NO	NO	NO	No effect.	Data to be loaded is not compressed.

Note:

1. A compression dictionary can be created for the XML storage object of a table only if the XML columns are added to the table in Db2 Version 9.7 or later, or if the table is migrated using an online table move.
2. If **LOAD REPLACE KEEPDICTIONARY** operation is interrupted, load utility can recover after either **LOAD RESTART** or **LOAD TERMINATE** is issued. Existing XML storage object dictionary may not be preserved after recovery from interrupted **LOAD REPLACE KEEPDICTIONARY** operation. A new XML storage object dictionary will be created if **LOAD RESTART** is used

RESETDICTIONARY

This directive instructs **LOAD REPLACE** processing to build a new dictionary for the table data object, provided that the table COMPRESS attribute is YES.

If the COMPRESS attribute is NO and a dictionary was already present in the table it will be removed and no new dictionary will be inserted into the table.

This option is the default for column-organized tables.

For column-organized tables, you cannot specify **LOAD REPLACE ... RESETDICTIONARY** on a subset of database partitions (SQL27906N). If you include the **OUTPUT_DBPARTNUMS** option, all database partitions must be specified.

A compression dictionary can be built with just one user record. If the loaded data set size is zero and if there is a preexisting dictionary, the dictionary will not be preserved. The amount of data required to build a dictionary with this directive is not subject to the policies of ADC. For a summary, see the following table.

Table 13. LOAD REPLACE RESETDICTIONARY

Compress	Table row data dictionary exists	XML storage object dictionary exists ¹	Compression dictionary	Data compression
YES	YES	YES	Build new dictionaries ² . If the DATA CAPTURE CHANGES option is enabled on the CREATE TABLE or ALTER TABLE statements, the current table row data dictionary is kept (and referred to as the <i>historical compression dictionary</i>).	After dictionaries are built, remaining data to be loaded is subject to compression.
YES	YES	NO	Build new dictionaries ² . If the DATA CAPTURE CHANGES option is enabled on the CREATE TABLE or ALTER TABLE statements, the current table row data dictionary is kept (and referred to as the <i>historical compression dictionary</i>).	After dictionaries are built, remaining data to be loaded is subject to compression.
YES	NO	YES	Build new dictionaries.	After dictionaries are built, remaining data to be loaded is subject to compression.

Table 13. **LOAD REPLACE RESETDICTIONARY** (continued)

Compress	Table row data dictionary exists	XML storage object dictionary exists ¹	Compression dictionary	Data compression
YES	NO	NO	Build new dictionaries.	After dictionaries are built, remaining data to be loaded is subject to compression.
NO	YES	YES	Remove dictionaries.	Data to be loaded is not compressed.
NO	YES	NO	Remove table row data dictionary.	Data to be loaded is not compressed.
NO	NO	YES	Remove XML storage object dictionary.	Data to be loaded is not compressed.
NO	NO	NO	No effect.	All table data is not compressed.

Notes:

1. A compression dictionary can be created for the XML storage object of a table only if the XML columns are added to the table in Db2 Version 9.7 or later, or if the table is migrated using an online table move.
2. If a dictionary exists and the compression attribute is enabled, but there are no records to load into the table partition, a new dictionary cannot be built and the **RESETDICTIONARY** operation will not keep the existing dictionary.

RESETDICTIONARYONLY

This option creates a column compression dictionary that is based on the input file, without loading any rows. You can use this option to create the compression dictionary before you ingest any data by using SQL-based utilities.

This option is applicable to column-organized tables only.

You cannot specify **LOAD REPLACE ... RESETDICTIONARYONLY** on a subset of database partitions (SQL27906N). If you include the **OUTPUT_DBPARTNUMS** option, all database partitions must be specified.

TERMINATE

One of four modes under which the load utility can execute. Terminates a previously interrupted load operation, and rolls back the operation to the point in time at which it started, even if consistency points were passed. The states of any table spaces involved in the operation return to normal, and all table objects are made consistent (index objects might be marked as invalid, in which case index rebuild will automatically take place at next access). If the load operation being terminated is a **LOAD REPLACE**, the table will be truncated to an empty table after the **LOAD TERMINATE** operation. If the load operation being terminated is a **LOAD INSERT**, the table will retain all of its original records after the **LOAD TERMINATE** operation. For summary of dictionary management, see Table 3.

The **LOAD TERMINATE** option will not remove a backup pending state from table spaces.

RESTART

Restarts an interrupted load operation. The load operation automatically

continues from the last consistency point in the load, build, or delete phase. For summary of dictionary management, see Table 4.

The **RESTART** parameter is not supported for the following tables:

- Column-organized tables
- Random distribution tables using the random by generation method

To recover a table of this type after a failed load operation, use the **TERMINATE** or **REPLACE** parameter.

INTO *table-name*

Specifies the database table into which the data is to be loaded. This table cannot be a system table, a declared temporary table, or a created temporary table. An alias, or the fully qualified or unqualified table name can be specified. A qualified table name is in the form *schema.tablename*. If an unqualified table name is specified, the table will be qualified with the CURRENT SCHEMA.

If the database table contains implicitly hidden columns, you must specify whether data for the hidden columns is included in the load operation. Use one of the following methods to indicate if data for hidden columns is included:

- Use *insert-column* to explicitly specify the columns into which data is to be inserted.

```
db2 load from delfile1 of del
      insert into table1 (c1, c2, c3,...)
```

- Use one of the hidden column file type modifiers: specify **implicitlyhiddeninclude** when the input file contains data for the hidden columns, or **implicitlyhiddenmissing** when the input file does not.

```
db2 load from delfile1 of del modified by implicitlyhiddeninclude
      insert into table1
```

- Use the DB2_DMU_DEFAULT registry variable on the server-side to set the default behavior when data movement utilities encounter tables with implicitly hidden columns. Specify **IMPLICITLYHIDDENINCLUDE** when utilities assume that the implicitly hidden columns are included, or **IMPLICITLYHIDDENMISSING** when utilities assume that the implicitly hidden columns are not included.

```
db2set DB2_DMU_DEFAULT=IMPLICITLYHIDDENINCLUDE
db2 load from delfile1 of del insert into table1
```

insert-column

Specifies the table column into which the data is to be inserted.

The load utility cannot parse columns whose names contain one or more spaces. For example,

```
db2 load from delfile1 of del noheader
      method P (1, 2, 3, 4, 5, 6, 7, 8, 9)
      insert into table1 (BLOB1, S2, I3, Int 4, I5, I6, DT7, I8, TM9)
```

will fail because of the Int 4 column. The solution is to enclose such column names with double quotation marks:

```
db2 load from delfile1 of del noheader
      method P (1, 2, 3, 4, 5, 6, 7, 8, 9)
      insert into table1 (BLOB1, S2, I3, "Int 4", I5, I6, DT7, I8, TM9)
```

FOR EXCEPTION *table-name*

Specifies the exception table into which rows in error are copied. Any row that violates a unique index or a primary key index is copied. If you

specify an unqualified table name, the table name is qualified with the current schema. The table cannot be a column-organized table.

Information that is written to the exception table is *not* written to the dump file. In a partitioned database environment, an exception table must be defined for those database partitions on which the loading table is defined. The dump file, otherwise, contains rows that cannot be loaded because they are invalid or have syntax errors.

When loading XML data, using the **FOR EXCEPTION** clause to specify a load exception table is not supported in the following cases:

- When using label-based access control (LBAC).
- When loading data into a partitioned table.

NORANGEEXC

Indicates that if a row is rejected because of a range violation it will not be inserted into the exception table.

NOUNIQUEEXC

Indicates that if a row is rejected because it violates a unique constraint it will not be inserted into the exception table.

STATISTICS USE PROFILE

Instructs load to collect statistics during the load according to the profile defined for this table. This is the default for column-organized tables. The profile must be created before issuing the **LOAD** command. The profile is created by the **RUNSTATS** command. For row-organized tables, if the profile does not exist and this parameter is specified, a warning is returned and no statistics are collected. For column-organized tables, if the profile does not exist and this parameter is specified, the load utility uses the same default **RUNSTATS** command options as those that are used during an automatic runstats operation.

During load, distribution statistics are not collected for columns of type XML.

STATISTICS NO

Specifies that no statistics are to be collected, and that the statistics in the catalogs are not to be altered. This parameter is the default for row-organized tables.

COPY NO

Specifies that the table space in which the table resides will be placed in backup pending state if forward recovery is enabled (that is, if either **logarchmeth1** or **logarchmeth2** is set to a value other than OFF). The **COPY NO** option will also put the table space state into the Load in Progress table space state. This is a transient state that will disappear when the load completes or fails. The data in any table in the table space cannot be updated or deleted until a table space backup or a full database backup is made. However, it is possible to access the data in any table by using the **SELECT** statement.

LOAD with **COPY NO** on a recoverable database leaves the table spaces in a backup pending state. For example, performing a **LOAD** with **COPY NO** and **INDEXING MODE DEFERRED** will leave indexes needing a refresh. Certain queries on the table might require an index scan and will not succeed until the indexes are refreshed. The index cannot be refreshed if it resides in a table space which is in the backup pending state. In that case, access to the table will not be allowed until a backup is taken. Index refresh is done automatically by the database when the index is accessed by a query. If

one of **COPY NO**, **COPY YES**, or **NONRECOVERABLE** is not specified, and the database is recoverable (**logarchmeth1** or **logarchmeth2** is set to value other than OFF), then **COPY NO** is the default.

COPY YES

Saves a copy of the loaded data. This parameter is invalid if forward recovery is disabled.

USE TSM

Specifies that the copy will be stored using IBM Tivoli Storage Manager.

OPEN *num-sess* SESSIONS

The number of I/O sessions to be used with TSM or the vendor product. The default value is 1.

TO *device/directory*

Specifies the device or directory on which the copy image will be created.

The directory can be on a remote storage, such as IBM® SoftLayer® Object Storage or Amazon Simple Storage Service (S3), and is being accessed using a storage access alias. Local staging space is required to temporarily store the copy image that is to be transferred to the remote storage server. The maximum copy image size for the remote storage is currently supported at 5GB.

LOAD *lib-name*

The name of the shared library (DLL on Windows operating systems) containing the vendor backup and restore I/O functions to be used. It can contain the full path. If the full path is not given, it will default to the path where the user exit programs reside.

NONRECOVERABLE

Specifies that the load transaction is to be marked as unrecoverable and that it will not be possible to recover it by a subsequent roll forward action. The roll forward utility will skip the transaction and will mark the table into which data was being loaded as “invalid”. The utility will also ignore any subsequent transactions against that table. After the roll forward operation is completed, such a table can only be dropped or restored from a backup (full or table space) taken after a commit point following the completion of the nonrecoverable load operation.

With this option, table spaces are not put in backup pending state following the load operation, and a copy of the loaded data does not have to be made during the load operation. If one of **COPY NO**, **COPY YES**, or **NONRECOVERABLE** is not specified, and the database is not recoverable (**logarchmeth1** and **logarchmeth2** are both set to OFF), then **NONRECOVERABLE** is the default.

WITHOUT PROMPTING

Specifies that the list of data files contains all the files that are to be loaded, and that the devices or directories listed are sufficient for the entire load operation. If a continuation input file is not found, or the copy targets are filled before the load operation finishes, the load operation will fail, and the table will remain in load pending state.

If this option is not specified, and the tape device encounters an end of tape for the copy image, or the last item listed is a tape device, the user is prompted for a new tape on that device.

DATA BUFFER *buffer-size*

Specifies the number of 4 KB pages (regardless of the degree of parallelism) to use as buffered space for transferring data within the utility. If the value specified is less than the algorithmic minimum, the minimum required resource is used, and no warning is returned.

This memory is allocated directly from the utility heap, whose size can be modified through the **util_heap_sz** database configuration parameter. Beginning in version 9.5, the value of the DATA BUFFER option of the **LOAD** command can temporarily exceed **util_heap_sz** if more memory is available in the system. In this situation, the utility heap is dynamically increased as needed until the **database_memory** limit is reached. This memory will be released once the load operation completes.

If a value is not specified, an intelligent default is calculated by the utility at run time. The default is based on a percentage of the free space available in the utility heap at the instantiation time of the loader, as well as some characteristics of the table.

SORT BUFFER *buffer-size*

This option specifies a value that overrides the **sortheap** database configuration parameter during a load operation. It is relevant only when loading tables with indexes and only when the **INDEXING MODE** parameter is not specified as DEFERRED. The value that is specified cannot exceed the value of **sortheap**. This parameter is useful for throttling the sort memory that is used when loading tables with many indexes without changing the value of **sortheap**, which would also affect general query processing.

CPU_PARALLELISM *n*

Specifies the number of processes or threads that the load utility will create for parsing, converting, and formatting records when building table objects. This parameter is designed to exploit the number of processes running per database partition. It is particularly useful when loading presorted data, because record order in the source data is preserved. If the value of this parameter is zero, or has not been specified, the load utility uses an intelligent default value (usually based on the number of CPUs available) at run time.

Note:

1. If this parameter is used with tables containing either LOB or LONG VARCHAR fields, its value becomes one, regardless of the number of system CPUs or the value specified by the user.
2. Specifying a small value for the **SAVECOUNT** parameter causes the loader to perform many more I/O operations to flush both data and table metadata. When **CPU_PARALLELISM** is greater than one, the flushing operations are asynchronous, permitting the loader to exploit the CPU. When **CPU_PARALLELISM** is set to one, the loader waits on I/O during consistency points. A load operation with **CPU_PARALLELISM** set to two, and **SAVECOUNT** set to 10 000, completes faster than the same operation with **CPU_PARALLELISM** set to one, even though there is only one CPU.

DISK_PARALLELISM *n*

Specifies the number of processes or threads that the load utility will create for writing data to the table space containers. If a value is not specified, the utility selects an intelligent default based on the number of table space containers and the characteristics of the table.

INDEXING MODE

Specifies whether the load utility is to rebuild indexes or to extend them incrementally. Valid values are:

AUTOSELECT

The load utility will automatically decide between REBUILD or INCREMENTAL mode. The decision is based on the amount of data being loaded and the depth of the index tree. Information relating to the depth of the index tree is stored in the index object. **RUNSTATS** is not required to populate this information. **AUTOSELECT** is the default indexing mode.

REBUILD

All indexes will be rebuilt. The utility must have sufficient resources to sort all index key parts for both old and appended table data.

If the LogIndexBuild database configuration parameter is turned on, the transaction log contains the image of each index page after it is created. If the LogIndexBuild database configuration parameter is turned off, only the allocation and initialization of each page is logged by the Index Manager (about 250 bytes per page approximately as opposed to the non-empty portion of each page).

INCREMENTAL

Indexes will be extended with new data. This approach consumes index free space. It only requires enough sort space to append index keys for the inserted records. This method is only supported in cases where the index object is valid and accessible at the start of a load operation (it is, for example, not valid immediately following a load operation in which the DEFERRED mode was specified). If this mode is specified, but not supported due to the state of the index, a warning is returned, and the load operation continues in REBUILD mode. Similarly, if a load restart operation is begun in the load build phase, INCREMENTAL mode is not supported.

If the LogIndexBuild database configuration parameter is turned on, the Db2 software generates the log records for the insertion of every key into the index as well as any page splits performed. If this parameter is turned off (which is common when not using HADR), the amount of index logging performed by the Index Manager depends on whether or not the ALLOW READ ACCESS option was specified. If the ALLOW READ ACCESS option is specified, the log record is generated including logs for page splits. If the ALLOW READ ACCESS option is not specified, no log record from the Index Manager is generated.

DEFERRED

The load utility will not attempt index creation if this mode is specified. Indexes will be marked as needing a refresh. The first access to such indexes that is unrelated to a load operation might force a rebuild, or indexes might be rebuilt when the database is restarted. This approach requires enough sort space for all key parts for the largest index. The total time subsequently taken for index construction is longer than that required in REBUILD mode. Therefore, when performing multiple load operations with deferred indexing, it is advisable (from a performance viewpoint) to let the

last load operation in the sequence perform an index rebuild, rather than allow indexes to be rebuilt at first non-load access.

Deferred indexing is only supported for tables with non-unique indexes, so that duplicate keys inserted during the load phase are not persistent after the load operation.

ALLOW NO ACCESS

Load will lock the target table for exclusive access during the load. The table state will be set to Load In Progress during the load. **ALLOW NO ACCESS** is the default behavior. It is the only valid option for **LOAD REPLACE**.

When there are constraints on the table, the table state will be set to Set Integrity Pending as well as Load In Progress. The SET INTEGRITY statement must be used to take the table out of Set Integrity Pending state.

ALLOW READ ACCESS

Load will lock the target table in a share mode. The table state will be set to both Load In Progress and Read Access. Readers can access the non-delta portion of the data while the table is being load. In other words, data that existed before the start of the load will be accessible by readers to the table, data that is being loaded is not available until the load is complete.

Important: Starting with Version 10.1 Fix Pack 1, the ALLOW READ ACCESS parameter is deprecated and might be removed in a future release. For more details, see “ALLOW READ ACCESS parameter in the LOAD command is deprecated” at .

The **ALLOW READ ACCESS** parameter is not supported for column-organized tables.

LOAD TERMINATE or **LOAD RESTART** of an **ALLOW READ ACCESS** load can use this parameter; **LOAD TERMINATE** or **LOAD RESTART** of an **ALLOW NO ACCESS** load cannot use this parameter. Furthermore, this option is not valid if the indexes on the target table are marked as requiring a rebuild.

When there are constraints on the table, the table state will be set to Set Integrity Pending as well as Load In Progress, and Read Access. At the end of the load, the table state Load In Progress will be removed but the table states Set Integrity Pending and Read Access will remain. The SET INTEGRITY statement must be used to take the table out of Set Integrity Pending. While the table is in Set Integrity Pending and Read Access states, the non-delta portion of the data is still accessible to readers, the new (delta) portion of the data will remain inaccessible until the SET INTEGRITY statement has completed. A user can perform multiple loads on the same table without issuing a SET INTEGRITY statement. Only the original (checked) data will remain visible, however, until the SET INTEGRITY statement is issued.

ALLOW READ ACCESS also supports the following modifiers:

USE *tablespace-name*

If the indexes are being rebuilt, a shadow copy of the index is built in table space *tablespace-name* and copied over to the original table space at the end of the load during an INDEX COPY PHASE. Only system temporary table spaces can be used with this option. If not specified then the shadow index will be created in the same table space as the index object. If the shadow copy is created in the same table space as the index object, the copy of the shadow index object

over the old index object is instantaneous. If the shadow copy is in a different table space from the index object a physical copy is performed. This could involve considerable I/O and time. The copy happens while the table is offline at the end of a load during the INDEX COPY PHASE.

Without this option the shadow index is built in the same table space as the original. Since both the original index and shadow index by default reside in the same table space simultaneously, there might be insufficient space to hold both indexes within one table space. Using this option ensures that you retain enough table space for the indexes.

This option is ignored if the user does not specify **INDEXING MODE REBUILD** or **INDEXING MODE AUTOSELECT**. This option will also be ignored if **INDEXING MODE AUTOSELECT** is chosen and load chooses to incrementally update the index.

FETCH_PARALLELISM YES | NO

When you perform a load from a cursor where the cursor is declared using the **DATABASE** keyword, or when using the API `sqlu_remotefetch_entry` media entry, and this option is set to YES, the load utility attempts to parallelize fetching from the remote data source if possible. Currently, you can parallelize data fetching only if the cursor's select-statement is of the simple form "SELECT * FROM <tablename>". If set to NO, no parallel fetching is performed. The default value is YES. For more information, see "Moving data using the CURSOR file type".

SET INTEGRITY PENDING CASCADE

If **LOAD** puts the table into Set Integrity Pending state, the **SET INTEGRITY PENDING CASCADE** option allows the user to specify whether or not Set Integrity Pending state of the loaded table is immediately cascaded to all descendants (including descendant foreign key tables, descendant immediate materialized query tables and descendant immediate staging tables).

IMMEDIATE

Indicates that Set Integrity Pending state is immediately extended to all descendant foreign key tables, descendant immediate materialized query tables and descendant staging tables. For a **LOAD INSERT** operation, Set Integrity Pending state is not extended to descendant foreign key tables even if the **IMMEDIATE** option is specified.

When the loaded table is later checked for constraint violations (using the IMMEDIATE CHECKED option of the SET INTEGRITY statement), descendant foreign key tables that were placed in Set Integrity Pending Read Access state will be put into Set Integrity Pending No Access state.

DEFERRED

Indicates that only the loaded table will be placed in the Set Integrity Pending state. The states of the descendant foreign key tables, descendant immediate materialized query tables and descendant immediate staging tables will remain unchanged.

descendant foreign key tables might later be implicitly placed in Set Integrity Pending state when their parent tables are checked for constraint violations (using the IMMEDIATE CHECKED option of the SET INTEGRITY statement). descendant immediate

materialized query tables and descendant immediate staging tables will be implicitly placed in Set Integrity Pending state when one of its underlying tables is checked for integrity violations. A query of a table that is in the Set Integrity Pending state might succeed if an eligible materialized query table that is not in the Set Integrity Pending state is accessed by the query instead of the specified table. A warning (SQLSTATE 01586) will be issued to indicate that descendant tables have been placed in Set Integrity Pending state. See the Notes section of the SET INTEGRITY statement for when these descendant tables will be put into Set Integrity Pending state.

If the **SET INTEGRITY PENDING CASCADE** option is not specified:

- Only the loaded table will be placed in Set Integrity Pending state. The state of descendant foreign key tables, descendant immediate materialized query tables and descendant immediate staging tables will remain unchanged, and can later be implicitly put into Set Integrity Pending state when the loaded table is checked for constraint violations.

If **LOAD** does not put the target table into Set Integrity Pending state, the **SET INTEGRITY PENDING CASCADE** option is ignored.

LOCK WITH FORCE

The utility acquires various locks including table locks in the process of loading. Rather than wait, and possibly timeout, when acquiring a lock, this option allows load to force off other applications that hold conflicting locks on the target table. Applications holding conflicting locks on the system catalog tables will not be forced off by the load utility. Forced applications will roll back and release the locks the load utility needs. The load utility can then proceed. This option requires the same authority as the **FORCE APPLICATIONS** command (SYSADM, SYSCTRL, or SYSMAINT).

ALLOW NO ACCESS loads might force applications holding conflicting locks at the start of the load operation. At the start of the load operation, the utility can force applications that are attempting to either query or modify the table.

ALLOW READ ACCESS loads can force applications holding conflicting locks at the start or end of the load operation. At the start of the load operation, the load utility can force applications that are attempting to modify the table. At the end of the load operation, the load utility can force applications that are attempting to either query or modify the table.

SOURCEUSEREXIT *executable*

Specifies the name of an executable file that is called to feed data into the utility.

The **SOURCEUSEREXIT** parameter is not supported for column-organized tables.

REDIRECT

INPUT FROM

BUFFER *input-buffer*

The stream of bytes specified in *input-buffer* is passed into the STDIN file descriptor of the process executing the given executable.

FILE *input-file*

The contents of this client-side file are passed into the STDIN file descriptor of the process executing the given executable.

OUTPUT TO**FILE *output-file***

The STDOUT and STDERR file descriptors are captured to the fully qualified server-side file specified.

PARALLELIZE

Increases the throughput of data coming into the load utility by invoking multiple user exit processes simultaneously. This option is only applicable in multi-partition database environments and is ignored in single-partition database environments.

For more information, see “Moving data using a customized application (user exit)”.

PARTITIONED DB CONFIG *partitioned-db-option*

Allows you to execute a load into a table distributed across multiple database partitions. The **PARTITIONED DB CONFIG** parameter allows you to specify partitioned database-specific configuration options. The *partitioned-db-option* values can be any of the following options:

```

PART_FILE_LOCATION x
OUTPUT_DBPARTNUMS x
PARTITIONING_DBPARTNUMS x
MODE x
MAX_NUM_PART_AGENTS x
ISOLATE_PART_ERRS x
STATUS_INTERVAL x
PORT_RANGE x
CHECK_TRUNCATION
MAP_FILE_INPUT x
MAP_FILE_OUTPUT x
TRACE x
NEWLINE
DISTFILE x
OMIT_HEADER
RUN_STAT_DBPARTNUM x

```

Detailed descriptions of these options are provided in “Load configuration options for partitioned database environments”.

RESTARTCOUNT

Deprecated.

USING *directory*

Deprecated.

Example 1

TABLE1 has 5 columns:

- COL1 VARCHAR 20 NOT NULL WITH DEFAULT
- COL2 SMALLINT
- COL3 CHAR 4
- COL4 CHAR 2 NOT NULL WITH DEFAULT
- COL5 CHAR 2 NOT NULL

ASCFILE1 has 7 elements:

- ELE1 positions 01 to 20
- ELE2 positions 21 to 22
- ELE3 positions 23 to 23
- ELE4 positions 24 to 27
- ELE5 positions 28 to 31
- ELE6 positions 32 to 32
- ELE7 positions 33 to 40

Data Records:

```
1...5....10....15....20....25....30....35....40
Test data 1      XXN 123abcdN
Test data 2 and 3  QQY   wxyzN
Test data 4,5 and 6 WWN6789   Y
```

The following command loads the table from the file:

```
db2 load from ascfile1 of asc modified by striptblanks reclen=40
method L (1 20, 21 22, 24 27, 28 31)
null indicators (0,0,23,32)
insert into table1 (col1, col5, col2, col3)
```

Note:

1. The specification of **striptblanks** in the **MODIFIED BY** parameter forces the truncation of blanks in VARCHAR columns (COL1, for example, which is 11, 17 and 19 bytes long, in rows 1, 2 and 3).
2. The specification of **reclen=40** in the **MODIFIED BY** parameter indicates that there is no newline character at the end of each input record, and that each record is 40 bytes long. The last 8 bytes are not used to load the table.
3. Since COL4 is not provided in the input file, it will be inserted into TABLE1 with its default value (it is defined NOT NULL WITH DEFAULT).
4. Positions 23 and 32 are used to indicate whether COL2 and COL3 of TABLE1 will be loaded NULL for a given row. If there is a Y in the column's null indicator position for a given record, the column will be NULL. If there is an N, the data values in the column's data positions of the input record (as defined in L(.....)) are used as the source of column data for the row. In this example, neither column in row 1 is NULL; COL2 in row 2 is NULL; and COL3 in row 3 is NULL.
5. In this example, the NULL INDICATORS for COL1 and COL5 are specified as 0 (zero), indicating that the data is not nullable.
6. The NULL INDICATOR for a given column can be anywhere in the input record, but the position must be specified, and the Y or N values must be supplied.

Example 2 (Loading LOBs from Files)

TABLE1 has 3 columns:

- COL1 CHAR 4 NOT NULL WITH DEFAULT
- LOB1 LOB
- LOB2 LOB

ASCFILE1 has 3 elements:

- ELE1 positions 01 to 04

- ELE2 positions 06 to 13
- ELE3 positions 15 to 22

The following files reside in either /u/user1 or /u/user1/bin:

- ASCFILE2 has LOB data
- ASCFILE3 has LOB data
- ASCFILE4 has LOB data
- ASCFILE5 has LOB data
- ASCFILE6 has LOB data
- ASCFILE7 has LOB data

Data Records in ASCFILE1:

```
1...5....10...15...20...25...30.
REC1 ASCFILE2 ASCFILE3
REC2 ASCFILE4 ASCFILE5
REC3 ASCFILE6 ASCFILE7
```

The following command loads the table from the file:

```
db2 load from ascfile1 of asc
      lobs from /u/user1, /u/user1/bin
      modified by lobsinfile reclen=22
      method L (1 4, 6 13, 15 22)
      insert into table1
```

Note:

1. The specification of **lobsinfile** in the **MODIFIED BY** parameter tells the loader that all LOB data is to be loaded from files.
2. The specification of **reclen=22** in the **MODIFIED BY** parameter indicates that there is no newline character at the end of each input record, and that each record is 22 bytes long.
3. LOB data is contained in 6 files, ASCFILE2 through ASCFILE7. Each file contains the data that will be used to load a LOB column for a specific row. The relationship between LOBs and other data is specified in ASCFILE1. The first record of this file tells the loader to place REC1 in COL1 of row 1. The contents of ASCFILE2 will be used to load LOB1 of row 1, and the contents of ASCFILE3 will be used to load LOB2 of row 1. Similarly, ASCFILE4 and ASCFILE5 will be used to load LOB1 and LOB2 of row 2, and ASCFILE6 and ASCFILE7 will be used to load the LOBs of row 3.
4. The LOBS FROM parameter contains 2 paths that will be searched for the named LOB files when those files are required by the loader.
5. To load LOBs directly from ASCFILE1 (a nondelimited ASCII file), without the **lobsinfile** modifier, the following rules must be observed:
 - The total length of any record, including LOBs, cannot exceed 32 KB.
 - LOB fields in the input records must be of fixed length, and LOB data padded with blanks as necessary.
 - The **striptblanks** modifier must be specified, so that the trailing blanks used to pad LOBs can be removed as the LOBs are inserted into the database.

Example 3 (Using Dump Files)

Table FRIENDS is defined as:

```
table friends "( c1 INT NOT NULL, c2 INT, c3 CHAR(8) )"
```

If an attempt is made to load the following data records into this table,

```
23, 24, bobby
, 45, john
4,, mary
```

the second row is rejected because the first INT is NULL, and the column definition specifies NOT NULL. Columns which contain initial characters that are not consistent with the DEL format will generate an error, and the record will be rejected. Such records can be written to a dump file.

DEL data appearing in a column outside of character delimiters is ignored, but does generate a warning. For example:

```
22,34,"bob"
24,55,"sam" sdf
```

The utility will load "sam" in the third column of the table, and the characters "sdf" will be flagged in a warning. The record is not rejected. Another example:

```
22 3, 34,"bob"
```

The utility will load 22,34,"bob", and generate a warning that some data in column one following the 22 was ignored. The record is not rejected.

Example 4 (Loading a Table with an Identity Column)

TABLE1 has 4 columns:

- C1 VARCHAR(30)
- C2 INT GENERATED BY DEFAULT AS IDENTITY
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 is the same as TABLE1, except that C2 is a GENERATED ALWAYS identity column.

Data records in DATAFILE1 (DEL format):

```
"Liszt"
"Hummel",,187.43, H
"Grieg",100, 66.34, G
"Satie",101, 818.23, I
```

Data records in DATAFILE2 (DEL format):

```
"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

Note:

1. The following command generates identity values for rows 1 and 2, since no identity values are supplied in DATAFILE1 for those rows. However, row 3 is assigned the user-supplied identity value of 100 and row 4 is assigned the user-supplied identity value of 101.

```
db2 load from datafile1.del of del replace into table1
```

2. To load DATAFILE1 into TABLE1 so that identity values are generated for all rows, issue one of the following commands:

- ```
db2 load from datafile1.del of del method P(1, 3, 4)
 replace into table1 (c1, c3, c4)
db2 load from datafile1.del of del modified by identityignore
 replace into table1
```
3. To load DATAFILE2 into TABLE1 so that identity values are generated for each row, issue one of the following commands:
 

```
db2 load from datafile2.del of del replace into table1 (c1, c3, c4)
db2 load from datafile2.del of del modified by identitymissing
 replace into table1
```
  4. To load DATAFILE1 into TABLE2 so that the identity values of 100 and 101 are assigned to rows 3 and 4, issue the following command:
 

```
db2 load from datafile1.del of del modified by identityoverride
 replace into table2
```

In this case, rows 1 and 2 will be rejected, because the utility has been instructed to override system-generated identity values in favor of user-supplied values. If user-supplied values are not present, however, the row must be rejected, because identity columns are implicitly not NULL.

5. If DATAFILE1 is loaded into TABLE2 without using any of the identity-related file type modifiers, rows 1 and 2 will be loaded, but rows 3 and 4 will be rejected, because they supply their own non-NULL values, and the identity column is GENERATED ALWAYS.

#### Example 5 (Loading a Table with a Row Change Timestamp Column)

TABLE1 has 4 columns:

- C1 VARCHAR(30)
- C2 ROW CHANGE TIMESTAMP GENERATED BY DEFAULT
- C3 DECIMAL(7,2)
- C4 CHAR(1)

TABLE2 is the same as TABLE1, except that C2 is a GENERATED ALWAYS column.

Data records in DATAFILE1 (DEL format):

```
"Liszt"
"Hummel",,187.43, H
"Grieg", 2006-05-23-15.55.53.209971, 66.34, G
"Satie", 2006-05-22-19.34.14.947681, 818.23, I
```

Data records in DATAFILE2 (DEL format):

```
"Liszt", 74.49, A
"Hummel", 0.01, H
"Grieg", 66.34, G
"Satie", 818.23, I
```

#### Note:

1. The following command generates ROW CHANGE TIMESTAMP values for rows 1 and 2, since no ROW CHANGE TIMESTAMP values are supplied in DATAFILE1 for those rows. Rows 3 and 4, however, are assigned the user-supplied ROW CHANGE TIMESTAMP values of 2006-05-23-15.55.53.209971 and 2006-05-22-19.34.14.947681.
 

```
db2 load from datafile1.del of del replace
 into table1
```

2. To load DATAFILE1 into TABLE1 so that ROW CHANGE TIMESTAMP values are generated for all rows, issue one of the following commands:

```
db2 load from datafile1.del of del method P(1, 3, 4) replace into table1
(c1, c3, c4)
```

```
db2 load from datafile1.del of del modified by rowchangetimestampignore
replace into table1
```

3. To load DATAFILE2 into TABLE1 so that ROW CHANGE TIMESTAMP values are generated for each row, issue one of the following commands:

```
db2 load from datafile2.del of del replace into table1 (c1, c3, c4)
```

```
db2 load from datafile2.del of del modified by rowchangetimestampmissing
replace into table1
```

4. To load DATAFILE1 into TABLE2 so that the ROW CHANGE TIMESTAMP values of 2006-05-23-15.55.53.209971 and 2006-05-22-19.34.14.947681 are assigned to rows 3 and 4, issue the following command:

```
db2 load from datafile1.del of del modified by rowchangetimestampoverride
replace into table2
```

In this case, rows 1 and 2 will be rejected, because the utility has been instructed to override system-generated ROW CHANGE TIMESTAMP values in favor of user-supplied values. If user-supplied values are not present, however, the row must be rejected, because row change timestamp columns are implicitly not NULL.

5. If DATAFILE1 is loaded into TABLE2 without using any of the ROW CHANGE related file type modifiers, rows 1 and 2 will be loaded, but rows 3 and 4 will be rejected, because they supply their own non-NULL values, and the row change timestamp column is GENERATED ALWAYS.

#### Example 6 (Loading using the CURSOR file type)

Table ABC.TABLE1 has 3 columns:

```
ONE INT
TWO CHAR(10)
THREE DATE
```

Table ABC.TABLE2 has 3 columns:

```
ONE VARCHAR
TWO INT
THREE DATE
```

Executing the following commands will load all the data from ABC.TABLE1 into ABC.TABLE2:

```
db2 declare mycurs cursor for select two,one,three from abc.table1
db2 load from mycurs of cursor insert into abc.table2
```

If ABC.TABLE1 resides in a database different from the database ABC.TABLE2 is in, the DATABASE, USER, and USING options of the DECLARE CURSOR statement can be used to perform the load. For example, if ABC.TABLE1 resides in database DB1, and the user ID and password for DB1 are user1 and pwd1, executing the following commands will load all the data from ABC.TABLE1 into ABC.TABLE2:

```
db2 declare mycurs cursor database DB1 user user1 using pwd1
for select two,one,three from abc.table1
db2 load from mycurs of cursor insert into abc.table2
```

### Example : Loading XML data

The user has constructed a data file with XDS fields to describe the documents that are to be inserted into the table. It might appear like this :

```
1, "<XDS FIL='file1.xml' />"
2, "<XDS FIL='file2.xml' OFF='23' LEN='45' />"
```

For the first row, the XML document is identified by the file named file1.xml. Since the character delimiter is the double quote character, and double quotation marks exist inside the XDS, the double quotation marks contained within the XDS are doubled. For the second row, the XML document is identified by the file named file2.xml, and starts at byte offset 23, and is 45 bytes in length.

The user issues a load command without any parsing or validation options for the XML column, and the data is loaded successfully:

```
LOAD
FROM data.del of DEL INSERT INTO mytable
```

### Example : Loading XML data from CURSOR

Loading data from cursor is the same as with a regular relational column type. The user has two tables, T1 and T2, each of which consist of a single XML column named C1. To LOAD from T1 into T2, the user will first declare a cursor:

```
DECLARE
X1 CURSOR FOR SELECT C1 FROM T1;
```

Next, the user may issue a **LOAD** using the cursor type:

```
LOAD FROM X1 of
CURSOR INSERT INTO T2
```

Applying the XML specific **LOAD** options to the cursor type is the same as loading from a file.

### Examples of using the XMLVALIDATE clause

#### XMLVALIDATE USING XDS

- The user would like to validate according to the schema indicated in the XDS. If there are any XDS values without SCH attributes, these documents will be validated against the schema that is used in this clause.

```
XMLVALIDATE
USING XDS DEFAULT S1.SCHEMA_B
```

- The user would like to validate but IGNORE certain schemas that are mentioned in the XDS.

```
XMLVALIDATE USING XDS IGNORE S1.SCHEMA_C
```

- The user would like to validate but remap some of the schemas in the XDS.

```
XMLVALIDATE USING XDS MAP((S1.SCHEMA_A,
S2.SCHEMA_B), (S3.SCHEMA_C, S5.SCHEMA_E))
```

Given the previously mentioned XDS, any document with an SCH attribute of S1.SCHEMA\_A will be validated against S2.SCHEMA\_B. Also, any document with an SCH attribute of S3.SCHEMA\_C will be validated against S5.SCHEMA\_E.

- The user would like to use a combination of the **DEFAULT**, **IGNORE**, and **MAP** options:

```
XMLVALIDATE USING XDS
 DEFAULT S8.SCHEMA_H
 IGNORE (S9.SCHEMA_I, S10.SCHEMA_J)
 MAP ((S1.SCHEMA_A, S2.SCHEMA_B), (S3.SCHEMA_C, S5.SCHEMA_E),
 (S6.SCHEMA_F, S3.SCHEMA_C), (S4.SCHEMA_D, S7.SCHEMA_G))
```

- For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The XML schema with SQL identifier "S8.SCHEMA\_H" is used to validate the document in file `xmlfile.001.xml`, since "S8.SCHEMA\_H" was specified as the default schema to use.

- For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' OFF='10' LEN='500' SCH='S10.SCHEMA_J' />
```

No schema validation occurs for the document in file `xmlfile.002.xml`, since although the XDS specifies "S10.SCHEMA\_J" as the schema to use, that schema is part of the **IGNORE** clause. The document contents can be found at byte offset 10 in the file (meaning the 11th byte), and is 500 bytes long.

- For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.003.xml' SCH='S6.SCHEMA_F' />
```

The XML schema with SQL identifier "S3.SCHEMA\_C" is used to validate the document in file `xmlfile.003.xml`. This is because the **MAP** clause specifies that schema "S6.SCHEMA\_F" should be mapped to schema "S3.SCHEMA\_C". Further mapping does not take place, therefore the mapping of schema "S3.SCHEMA\_C" to schema "S5.SCHEMA\_E" does not apply in this case.

- For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.004.xml' SCH='S11.SCHEMA_K' />
```

The XML schema with SQL identifier "S11.SCHEMA\_K" is used to validate the document in file `xmlfile.004.xml`. None of the **DEFAULT**, **IGNORE**, or **MAP** specifications apply in this case.

## XMLVALIDATE USING SCHEMA

The user wants to validate all XML documents according to a single SCHEMA. For the following **XMLVALIDATE** clause:

```
XMLVALIDATE USING SCHEMA S2.SCHEMA_B
```

- For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The document in file `xmlfile.001.xml` is validated using the XML schema with SQL identifier "S2.SCHEMA\_B".

- For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' SCH='S1.SCHEMA_A' />
```

The document in file `xmlfile.002.xml` is validated using the XML schema with SQL identifier "S2.SCHEMA\_B". The SCH attribute is ignored, since validation is being performed using a schema specified by the **USING SCHEMA** clause.

## XMLVALIDATE USING SCHEMALOCATION HINTS

The user would like to validate against schema information located within the document itself. For the following **XMLVALIDATE** clause:

```
XMLVALIDATE
USING SCHEMALOCATION HINTS
```

- For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.001.xml' />
```

The XML schema used is determined by the schemaLocation attribute in the document contents, and no validation would occur if one is not present.

- For an XML column that contains the following XDS:

```
<XDS FIL='xmlfile.002.xml' SCH='S1.SCHEMA_A' />
```

The XML schema used is determined by the schemaLocation attribute in the document contents, and no validation would occur if one is not present. The SCH attribute is ignored, since validation is being performed using **SCHEMALOCATION HINTS**.

### Usage notes

- Data is loaded in the sequence that appears in the input file. If a particular sequence is desired, the data should be sorted before a load is attempted. If preservation of the source data order is not required, consider using the **ANYORDER** file type modifier, described in the following “File type modifiers for the load utility” section.
- The load utility builds indexes based on existing definitions. The exception tables are used to handle duplicates on unique keys. The utility does not enforce referential integrity, perform constraints checking, or update materialized query tables that are dependent on the tables being loaded. Tables that include referential or check constraints are placed in Set Integrity Pending state. Summary tables that are defined with REFRESH IMMEDIATE, and that are dependent on tables being loaded, are also placed in Set Integrity Pending state. Issue the SET INTEGRITY statement to take the tables out of Set Integrity Pending state. Load operations cannot be carried out on replicated materialized query tables.
- If a clustering index exists on the table, the data should be sorted on the clustering index before loading. Data does not need to be sorted before loading into a multidimensional clustering (MDC) table, however.
- If you specify an exception table when loading into a protected table, any rows that are protected by invalid security labels will be sent to that table. This might allow users that have access to the exception table to access to data that they would not normally be authorized to access. For better security be careful who you grant exception table access to, delete each row as soon as it is repaired and copied to the table being loaded, and drop the exception table as soon as you are done with it.
- Security labels in their internal format might contain newline characters. If you load the file using the DEL file format, those newline characters can be mistaken for delimiters. If you have this problem use the older default priority for delimiters by specifying the **delprioritychar** file type modifier in the **LOAD** command.
- For performing a load using the CURSOR file type where the DATABASE keyword was specified during the DECLARE CURSOR statement, the user ID and password used to authenticate against the database currently connected to (for the load) will be used to authenticate against the source database (specified by the DATABASE option of the DECLARE CURSOR statement). If no user ID

or password was specified for the connection to the loading database, a user ID and password for the source database must be specified during the **DECLARE CURSOR** statement.

- Loading a multiple-part PC/IXF file whose individual parts are copied from a Windows system to an AIX system is supported. The names of all the files must be specified in the **LOAD** command. For example, **LOAD FROM DATA.IXF, DATA.002 OF IXF INSERT INTO TABLE1**. Loading to the Windows operating system from logically split PC/IXF files is not supported.
- When restarting a failed **LOAD**, the behavior will follow the existing behavior in that the **BUILD** phase will be forced to use the **REBUILD** mode for indexes.
- The Load utility might generate a very large copy of the image file when the **COPY YES** option is used. This behavior is expected when the **LOAD** command writes out an entire buffer of data to the copy image for every LOB/LF column value that is loaded. The buffer is an internal object, and its size is determined by several internal and external factors. Typically, the buffer size is between 68KB and a few hundred KB.
- Loading XML documents between databases is not supported and returns error message SQL1407N.
- The **LOAD** utility does not support loading into tables that contain columns that reference fenced procedures. If you issue the **LOAD** command on such table, you will receive error message SQL1376N. To work around this restriction, you can redefine the routine to be unfenced, or use the import utility.
- If a table contains a generated column expression in which the user-defined function is a compiled compound SQL, you can use the **LOAD** utility only with the **generatedoverride** file type modifier to insert data into the table. You can also use the import utility to insert data into these tables.
- If the database table contains implicitly hidden columns, you must specify whether data for the hidden columns is included in the load operation.
- The **IMPORT** utility does not match the number of columns in a table and the number of fields in a data file. The utility checks for a sufficient amount of data in the data file and if a row in the data file does not contain sufficient columns of data, the row may either be rejected with a warning message if the corresponding table columns without data are defined as **NOT NULL**, or be inserted successfully without a warning message if the corresponding table columns are defined as **NULL**. Alternatively, if a row contains a higher number of columns than required, the sufficient number of columns are processed while the remaining columns of data are omitted and no warning message is given.
- The **STATISTICS** options only work for the **LOAD REPLACE** option and do not work for other **LOAD** command options.
- When **LOAD** utility with **COPY YES** option is used, and the table contains LOB columns, **LOAD** always enforces **COMPACT YES** behavior even when the LOB column is defined with **COMPACT NOT**.

### Summary of **LOAD TERMINATE** and **LOAD RESTART** dictionary management

The following chart summarizes the compression dictionary management behavior for **LOAD** processing under the **TERMINATE** directive.

Table 14. **LOAD TERMINATE** dictionary management

Table COMPRESS attribute	Does table row data dictionary existed before LOAD?	XML storage object dictionary exists before LOAD <sup>1</sup>	TERMINATE: LOAD REPLACE KEEPDICTIONARY or LOAD INSERT	TERMINATE: LOAD REPLACE RESETDICTIONARY
YES	YES	YES	Keep existing dictionaries.	Neither dictionary is kept. <sup>2</sup>
YES	YES	NO	Keep existing dictionary.	Nothing is kept. <sup>2</sup>
YES	NO	YES	Keep existing dictionary.	Nothing is kept.
YES	NO	NO	Nothing is kept.	Nothing is kept.
NO	YES	YES	Keep existing dictionaries.	Nothing is kept.
NO	YES	NO	Keep existing dictionary.	Nothing is kept.
NO	NO	YES	Keep existing dictionary.	Nothing is kept.
NO	NO	NO	Do nothing.	Do nothing.

**Note:**

1. A compression dictionary can be created for the XML storage object of a table only if the XML columns are added to the table in Db2 Version 9.7 or later, or if the table is migrated using an online table move.
2. In the special case that the table has data capture enabled, the table row data dictionary is kept.

**LOAD RESTART** truncates a table up to the last consistency point reached. As part of **LOAD RESTART** processing, a compression dictionary will exist in the table if it was present in the table at the time the last **LOAD** consistency point was taken. In that case, **LOAD RESTART** will not create a new dictionary. For a summary of the possible conditions, see Table 4.

Table 15. **LOAD RESTART** dictionary management

Table COMPRESS Attribute	Table row data dictionary exist before LOAD consistency point? <sup>1</sup>	XML Storage object dictionary existed before last LOAD? <sup>2</sup>	RESTART: LOAD REPLACE KEEPDICTIONARY or LOAD INSERT	RESTART: LOAD REPLACE RESETDICTIONARY
YES	YES	YES	Keep existing dictionaries.	Keep existing dictionaries.
YES	YES	NO	Keep existing table row data dictionary and build XML dictionary subject to ADC.	Keep existing table row data dictionary and build XML dictionary.
YES	NO	YES	Build table row data dictionary subject to ADC. Keep existing XML dictionary.	Build table row data dictionary. Keep existing XML dictionary.
YES	NO	NO	Build table row data and XML dictionaries subject to ADC.	Build table row data and XML dictionaries.
NO	YES	YES	Keep existing dictionaries.	Remove existing dictionaries.
NO	YES	NO	Keep existing table row data dictionary.	Remove existing table row data dictionary.

Table 15. LOAD RESTART dictionary management (continued)

Table COMPRESS Attribute	Table row data dictionary exist before LOAD consistency point? <sup>1</sup>	XML Storage object dictionary existed before last LOAD? <sup>2</sup>	RESTART: LOAD REPLACE KEEPDICTIONARY or LOAD INSERT	RESTART: LOAD REPLACE RESETDICTIONARY
NO	NO	YES	Keep existing XML dictionary.	Remove existing XML dictionary.
NO	NO	NO	Do nothing.	Do nothing.

**Notes:**

1. The **SAVECOUNT** option is not allowed when loading XML data, load operations that fail during the load phase restart from the beginning of the operation.
2. A compression dictionary can be created for the XML storage object of a table only if the XML columns are added to the table in Db2 Version 9.7 or later, or if the table is migrated using an online table move.

**File type modifiers for the load utility**

Table 16. Valid file type modifiers for the load utility: All file formats

Modifier	Description
<b>anyorder</b>	This modifier specifies that the preservation of the source data order is not required. Using this modifier yields significant additional performance benefits on SMP systems. Use this modifier with the <b>cpu_parallelism</b> parameter. If the value of the <b>cpu_parallelism</b> parameter is 1, this modifier is ignored. This modifier is not supported if the value of the <b>SAVECOUNT</b> parameter is greater than 0, because crash recovery after a consistency point requires that data be loaded in sequence. This modifier is implicitly turned on for all load operations for column-organized tables, multidimensional clustering (MDC) tables, and range-partitioned tables.
<b>cdeanalyzefrequency=x</b>	<i>x</i> is an integer between 0 - 99 inclusive. This value controls how much data is sampled in the ANALYZE phase to produce a compression dictionary. In a massively parallel processing (MPP), the sampling size is not aggregated across members. The ANALYZE phase is stopped when the first member reaches max.
<b>generatedignore</b>	This modifier informs the load utility that data for all generated columns is present in the data file but should be ignored. This results in all generated column values being generated by the utility. This modifier cannot be used with either the <b>generatedmissing</b> or the <b>generatedoverride</b> modifier.
<b>generatedmissing</b>	If this modifier is specified, the utility assumes that the input data file contains no data for the generated column (not even NULLs). This results in all generated column values being generated by the utility. This modifier cannot be used with either the <b>generatedignore</b> or the <b>generatedoverride</b> modifier.

Table 16. Valid file type modifiers for the load utility: All file formats (continued)

Modifier	Description
<b>generatedoverride</b>	<p>This modifier instructs the load utility to accept user-supplied data for all generated columns in the table (contrary to the normal rules for these types of columns). This is useful when migrating data from another database system, or when loading a table from data that was recovered using the <b>RECOVER DROPPED TABLE</b> option on the <b>ROLLFORWARD DATABASE</b> command. When this modifier is used, any rows with no data or NULL data for a non-nullable generated column will be rejected (SQL3116W). When this modifier is used, the table will be placed in Set Integrity Pending state. To take the table out of Set Integrity Pending state without verifying the user-supplied values, issue the following command after the load operation:</p> <pre>SET INTEGRITY FOR <i>table-name</i> GENERATED COLUMN IMMEDIATE UNCHECKED</pre> <p>To take the table out of Set Integrity Pending state and force verification of the user-supplied values, issue the following command after the load operation:</p> <pre>SET INTEGRITY FOR <i>table-name</i> IMMEDIATE CHECKED.</pre> <p>When this modifier is specified and there is a generated column in any of the partitioning keys, dimension keys or distribution keys, then the <b>LOAD</b> command automatically converts the modifier to <b>generatedignore</b> and proceeds with the load. This has the effect of regenerating all the generated column values.</p> <p>This modifier cannot be used with either the <b>generatedmissing</b> or the <b>generatedignore</b> modifier.</p> <p>This modifier cannot be used for column-organized tables (SQLSTATE 42858).</p> <p>Random distribution tables that use the random by generation method have an internally generated column called the <b>RANDOM_DISTRIBUTION_KEY</b>. This modifier does not apply to that column, only to other generated columns in the table. Values for the <b>RANDOM_DISTRIBUTION_KEY</b> will always be regenerated.</p>
<b>identityignore</b>	<p>This modifier informs the load utility that data for the identity column is present in the data file but should be ignored. This results in all identity values being generated by the utility. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. This means that for GENERATED ALWAYS columns, no rows will be rejected. This modifier cannot be used with either the <b>identitymissing</b> or the <b>identityoverride</b> modifier.</p>
<b>identitymissing</b>	<p>If this modifier is specified, the utility assumes that the input data file contains no data for the identity column (not even NULLs), and will therefore generate a value for each row. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT identity columns. This modifier cannot be used with either the <b>identityignore</b> or the <b>identityoverride</b> modifier.</p>
<b>identityoverride</b>	<p>This modifier should be used only when an identity column defined as GENERATED ALWAYS is present in the table to be loaded. It instructs the utility to accept explicit, non-NULL data for such a column (contrary to the normal rules for these types of identity columns). This is useful when migrating data from another database system when the table must be defined as GENERATED ALWAYS, or when loading a table from data that was recovered using the <b>DROPPED TABLE RECOVERY</b> option on the <b>ROLLFORWARD DATABASE</b> command. When this modifier is used, any rows with no data or NULL data for the identity column will be rejected (SQL3116W). This modifier cannot be used with either the <b>identitymissing</b> or the <b>identityignore</b> modifier. The load utility will not attempt to maintain or verify the uniqueness of values in the table's identity column when this option is used.</p>

Table 16. Valid file type modifiers for the load utility: All file formats (continued)

Modifier	Description
<b>implicitlyhiddeninclude</b>	<p>If this modifier is specified, the utility assumes that the input data file contains data for the implicitly hidden columns and this data will also be loaded. This modifier cannot be used with the <b>implicitlyhiddenmissing</b> modifier. See the “Notes” on page 121 section for information about the precedence when multiple modifiers are specified.</p> <p>Random distribution tables that use the random by generation method have a hidden column as its distribution column called <b>RANDOM_DISTRIBUTION_KEY</b>. This modifier does not apply to that column, only to the other hidden columns in the table. The <b>RANDOM_DISTRIBUTION_KEY</b> will be treated as if it were missing.</p>
<b>implicitlyhiddenmissing</b>	<p>If this modifier is specified, the utility assumes that the input data file does not contain data for the implicitly hidden columns and the utility will generate values for those hidden columns. This modifier cannot be used with the <b>implicitlyhiddeninclude</b> modifier. See the “Notes” on page 121 section for information about the precedence when multiple modifiers are specified.</p>
<b>indexfreespace=<i>x</i></b>	<p><i>x</i> is an integer between 0 and 99 inclusive. The value is interpreted as the percentage of each index page that is to be left as free space when load rebuilds the index. Load with <b>INDEXING MODE INCREMENTAL</b> ignores this option. The first entry in a page is added without restriction; subsequent entries are added to maintain the percent free space threshold. The default value is the one used at CREATE INDEX time.</p> <p>This value takes precedence over the PCTFREE value specified in the CREATE INDEX statement. The <b>indexfreespace</b> option affects index leaf pages only.</p>
<b>lobsinfile</b>	<p><i>lob-path</i> specifies the path to the files containing LOB data. The ASC, DEL, or IXF load input files contain the names of the files having LOB data in the LOB column.</p> <p>This option is not supported in conjunction with the CURSOR filetype.</p> <p>The <b>LOBS FROM</b> clause specifies where the LOB files are located when the <b>lobsinfile</b> modifier is used. The <b>LOBS FROM</b> clause will implicitly activate the <b>lobsinfile</b> behavior. The <b>LOBS FROM</b> clause conveys to the <b>LOAD</b> utility the list of paths to search for the LOB files while loading the data.</p> <p>Each path contains at least one file that contains at least one LOB pointed to by a Lob Location Specifier (LLS) in the data file. The LLS is a string representation of the location of a LOB in a file stored in the LOB file path. The format of an LLS is <i>filename.ext.nnn.mmm/</i>, where <i>filename.ext</i> is the name of the file that contains the LOB, <i>nnn</i> is the offset in bytes of the LOB within the file, and <i>mmm</i> is the length of the LOB in bytes. For example, if the string db2exp.001.123.456/ is stored in the data file, the LOB is located at offset 123 in the file db2exp.001, and is 456 bytes long.</p> <p>To indicate a null LOB , enter the size as -1. If the size is specified as 0, it is treated as a 0 length LOB. For null LOBS with length of -1, the offset and the file name are ignored. For example, the LLS of a null LOB might be db2exp.001.7.-1/.</p>
<b>maxanalyze size=<i>x</i></b>	<p><i>x</i> is the size that has a value of &lt;Number&gt;&lt;Megabytes Gigabytes&gt;. The default size is 128 GB. <b>maxanalyze size</b> controls how much data is sampled in the ANALYZE phase to produce compression dictionary. In a massively parallel processing (MPP), the sampling size is not aggregated across members. The ANALYZE phase is stopped when first member reaches max.</p> <p><b>Note:</b> A value of 0 means unlimited (full size). Here is an example:</p> <p>modified by maxanalyze size=1G modified by maxanalyze size=100M</p>

Table 16. Valid file type modifiers for the load utility: All file formats (continued)

Modifier	Description
<b>noheader</b>	<p>Skips the header verification code (applicable only to load operations into tables that reside in a single-partition database partition group).</p> <p>If the default MPP load (mode PARTITION_AND_LOAD) is used against a table residing in a single-partition database partition group, the file is not expected to have a header. Thus the <b>noheader</b> modifier is not needed. If the LOAD_ONLY mode is used, the file is expected to have a header. The only circumstance in which you should need to use the <b>noheader</b> modifier is if you wanted to perform LOAD_ONLY operation using a file that does not have a header.</p>
<b>norowwarnings</b>	Suppresses all warnings about rejected rows.
<b>pagefreespace=<i>x</i></b>	<p><i>x</i> is an integer between 0 and 100 inclusive. The value is interpreted as the percentage of each data page that is to be left as free space. If the specified value is invalid because of the minimum row size, (for example, a row that is at least 3 000 bytes long, and an <i>x</i> value of 50), the row will be placed on a new page. If a value of 100 is specified, each row will reside on a new page. The PCTFREE value of a table determines the amount of free space designated per page. If a <b>pagefreespace</b> value on the load operation or a PCTFREE value on a table have not been set, the utility will fill up as much space as possible on each page. The value set by <b>pagefreespace</b> overrides the PCTFREE value specified for the table.</p>
<b>periodignore</b>	This modifier informs the load utility that data for the period columns is present in the data file but should be ignored. When this modifier is specified, all period column values are generated by the utility. This modifier cannot be used with the <b>periodmissing</b> modifier or the <b>periodoverride</b> modifier.
<b>periodmissing</b>	If this modifier is specified, the utility assumes that the input data file contains no data for the period columns. When this modifier is specified, all period column values are generated by the utility. This modifier cannot be used with the <b>periodignore</b> modifier or the <b>periodoverride</b> modifier.
<b>periodoverride</b>	This modifier instructs the load utility to accept user-supplied data for GENERATED ALWAYS AS ROW BEGIN and GENERATED ALWAYS AS ROW END columns in a system-period temporal table. This behavior is contrary to the normal rules for these types of columns. The modifier can be useful when you want to maintain history data and load data that includes time stamps into a system-period temporal table. When this modifier is used, any rows with no data or NULL data in a ROW BEGIN or ROW END column are rejected.
<b>rowchangetimestampignore</b>	<p>This modifier informs the load utility that data for the row change timestamp column is present in the data file but should be ignored. This results in all ROW CHANGE TIMESTAMPS being generated by the utility. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT columns. This means that for GENERATED ALWAYS columns, no rows will be rejected. This modifier cannot be used with either the <b>rowchangetimestampmissing</b> or the <b>rowchangetimestampoverride</b> modifier.</p>
<b>rowchangetimestampmissing</b>	If this modifier is specified, the utility assumes that the input data file contains no data for the row change timestamp column (not even NULLs), and will therefore generate a value for each row. The behavior will be the same for both GENERATED ALWAYS and GENERATED BY DEFAULT columns. This modifier cannot be used with either the <b>rowchangetimestampignore</b> or the <b>rowchangetimestampoverride</b> modifier.

Table 16. Valid file type modifiers for the load utility: All file formats (continued)

Modifier	Description
<b>rowchangetimestampoverride</b>	<p>This modifier should be used only when a row change timestamp column defined as GENERATED ALWAYS is present in the table to be loaded. It instructs the utility to accept explicit, non-NULL data for such a column (contrary to the normal rules for these types of row change timestamp columns). This is useful when migrating data from another database system when the table must be defined as GENERATED ALWAYS, or when loading a table from data that was recovered using the <b>DROPPED TABLE RECOVERY</b> option on the <b>ROLLFORWARD DATABASE</b> command. When this modifier is used, any rows with no data or NULL data for the ROW CHANGE TIMESTAMP column will be rejected (SQL3116W). This modifier cannot be used with either the <b>rowchangetimestampmissing</b> or the <b>rowchangetimestampignore</b> modifier. The load utility will not attempt to maintain or verify the uniqueness of values in the table's row change timestamp column when this option is used.</p>
<b>seclabelchar</b>	<p>Indicates that security labels in the input source file are in the string format for security label values rather than in the default encoded numeric format. <b>LOAD</b> converts each security label into the internal format as it is loaded. If a string is not in the proper format the row is not loaded and a warning (SQLSTATE 01H53, SQLCODE SQL3242W) is returned. If the string does not represent a valid security label that is part of the security policy protecting the table then the row is not loaded and a warning (SQLSTATE 01H53, SQLCODE SQL3243W) is returned.</p> <p>This modifier cannot be specified if the <b>seclabelname</b> modifier is specified, otherwise the load fails and an error (SQLCODE SQL3525N) is returned.</p> <p>If you have a table consisting of a single DB2SECURITYLABEL column, the data file might look like this:</p> <pre>"CONFIDENTIAL:ALPHA:G2" "CONFIDENTIAL;SIGMA:G2" "TOP SECRET:ALPHA:G2"</pre> <p>To load or import this data, the <b>seclabelchar</b> file type modifier must be used:</p> <pre>LOAD FROM input.del OF DEL MODIFIED BY SECLABELCHAR INSERT INTO t1</pre>
<b>seclabelname</b>	<p>Indicates that security labels in the input source file are indicated by their name rather than the default encoded numeric format. <b>LOAD</b> will convert the name to the appropriate security label if it exists. If no security label exists with the indicated name for the security policy protecting the table the row is not loaded and a warning (SQLSTATE 01H53, SQLCODE SQL3244W) is returned.</p> <p>This modifier cannot be specified if the <b>seclabelchar</b> modifier is specified, otherwise the load fails and an error (SQLCODE SQL3525N) is returned.</p> <p>If you have a table consisting of a single DB2SECURITYLABEL column, the data file might consist of security label names similar to:</p> <pre>"LABEL1" "LABEL1" "LABEL2"</pre> <p>To load or import this data, the <b>seclabelname</b> file type modifier must be used:</p> <pre>LOAD FROM input.del OF DEL MODIFIED BY SECLABELNAME INSERT INTO t1</pre> <p><b>Note:</b> If the file type is ASC, any spaces following the name of the security label will be interpreted as being part of the name. To avoid this use the <b>striptblanks</b> file type modifier to make sure the spaces are removed.</p>

Table 16. Valid file type modifiers for the load utility: All file formats (continued)

Modifier	Description
<b>total freespace=<i>x</i></b>	<i>x</i> is an integer greater than or equal to 0. The value is interpreted as the percentage of the total pages in the table that is to be appended to the end of the table as free space. For example, if <i>x</i> is 20, and the table has 100 data pages after the data has been loaded, 20 additional empty pages will be appended. The total number of data pages for the table will be 120. The data pages total does not factor in the number of index pages in the table. This option does not affect the index object. If two loads are done with this option specified, the second load will not reuse the extra space appended to the end by the first load.
<b>transactionidignore</b>	This modifier informs the load utility that data for the TRANSACTION START ID column is present in the data file but should be ignored. When this modifier is specified, the value for the TRANSACTION START ID column is generated by the utility. This modifier cannot be used with the <b>transactionidmissing</b> modifier or the <b>transactionidoverride</b> modifier.
<b>transactionidmissing</b>	If this modifier is specified, the utility assumes that the input data file contains no data for the TRANSACTION START ID columns. When this modifier is specified, the value for the TRANSACTION START ID column is generated by the utility. This modifier cannot be used with the <b>transactionidignore</b> modifier or the <b>transactionidoverride</b> modifier.
<b>transactionidoverride</b>	This modifier instructs the load utility to accept user-supplied data for the GENERATED ALWAYS AS TRANSACTION START ID column in a system-period temporal table. This behavior is contrary to the normal rules for this type of column. When this modifier is used, any rows with no data or NULL data in a TRANSACTION START ID column are rejected.
<b>usedefaults</b>	<p>If a source column for a target table column has been specified, but it contains no data for one or more row instances, default values are loaded. Examples of missing data are:</p> <ul style="list-style-type: none"> <li>• For DEL files: two adjacent column delimiters (",,") or two adjacent column delimiters separated by an arbitrary number of spaces (" , ") are specified for a column value.</li> <li>• For DEL/ASC files: A row that does not have enough columns, or is not long enough for the original specification. For ASC files, NULL column values are not considered explicitly missing, and a default will not be substituted for NULL column values. NULL column values are represented by all space characters for numeric, date, time, and /timestamp columns, or by using the NULL INDICATOR for a column of any type to indicate the column is NULL.</li> </ul> <p>Without this option, if a source column contains no data for a row instance, one of the following occurs:</p> <ul style="list-style-type: none"> <li>• For DEL/ASC files: If the column is nullable, a NULL is loaded. If the column is not nullable, the utility rejects the row.</li> </ul>

Table 17. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL)

Modifier	Description
<b>codepage=x</b>	<p><i>x</i> is an ASCII character string. The value is interpreted as the code page of the data in the input data set. Converts character data (and numeric data specified in characters) from this code page to the database code page during the load operation.</p> <p>The following rules apply:</p> <ul style="list-style-type: none"> <li>• For pure DBCS (graphic), mixed DBCS, and EUC, delimiters are restricted to the range of x00 to x3F, inclusive.</li> <li>• For DEL data specified in an EBCDIC code page, the delimiters might not coincide with the shift-in and shift-out DBCS characters.</li> <li>• <b>nullindchar</b> must specify symbols included in the standard ASCII set between code points x20 and x7F, inclusive. This refers to ASCII symbols and code points. EBCDIC data can use the corresponding symbols, even though the code points will be different.</li> </ul> <p>This option is not supported in conjunction with the CURSOR filetype.</p>
<b>dateformat="x"</b>	<p><i>x</i> is the format of the date in the source file.<sup>1</sup> Valid date elements are:</p> <p>YYYY - Year (four digits ranging from 0000 - 9999)  M - Month (one or two digits ranging from 1 - 12)  MM - Month (two digits ranging from 01 - 12;  mutually exclusive with M)  D - Day (one or two digits ranging from 1 - 31)  DD - Day (two digits ranging from 01 - 31;  mutually exclusive with D)  DDD - Day of the year (three digits ranging  from 001 - 366; mutually exclusive  with other day or month elements)</p> <p>A default value of 1 is assigned for each element that is not specified. Some examples of date formats are:</p> <p>"D-M-YYYY"  "MM.DD.YYYY"  "YYYYDDD"</p>
<b>dumpfile = x</b>	<p><i>x</i> is the fully qualified (according to the server database partition) name of an exception file to which rejected rows are written. A maximum of 32 KB of data is written per record. The following section is an example that shows how to specify a dump file:</p> <pre>db2 load from data of del modified by dumpfile = /u/user/filename insert into table_name</pre> <p>The file will be created and owned by the instance owner. To override the default file permissions, use the <b>dumpfileaccessall</b> file type modifier.</p> <p><b>Note:</b></p> <ol style="list-style-type: none"> <li>1. In a partitioned database environment, the path should be local to the loading database partition, so that concurrently running load operations do not attempt to write to the same file.</li> <li>2. The contents of the file are written to disk in an asynchronous buffered mode. In the event of a failed or an interrupted load operation, the number of records committed to disk cannot be known with certainty, and consistency cannot be guaranteed after a <b>LOAD RESTART</b>. The file can only be assumed to be complete for a load operation that starts and completes in a single pass.</li> <li>3. If the specified file already exists, it will not be re-created, but it will be truncated.</li> </ol>

Table 17. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL) (continued)

Modifier	Description
<b>dumpfileaccessall</b>	<p>Grants read access to 'OTHERS' when a dump file is created.</p> <p>This file type modifier is only valid when:</p> <ol style="list-style-type: none"> <li>1. it is used in conjunction with <b>dumpfile</b> file type modifier</li> <li>2. the user has SELECT privilege on the load target table</li> <li>3. it is issued on a Db2 server database partition that resides on a UNIX operating system</li> </ol> <p>If the specified file already exists, its permissions will not be changed.</p>
<b>fastparse</b>	<p>Use with caution. Reduces syntax checking on user-supplied column values, and enhances performance. Tables are guaranteed to be architecturally correct (the utility performs sufficient data checking to prevent a segmentation violation or trap), however, the coherence of the data is not validated. Only use this option if you are certain that your data is coherent and correct. For example, if the user-supplied data contains an invalid timestamp column value of :1&gt;0-00-20-07.11.12.000000, this value is inserted into the table if <b>fastparse</b> is specified, and rejected if <b>fastparse</b> is not specified.</p>
<b>implieddecimal</b>	<p>The location of an implied decimal point is determined by the column definition; it is no longer assumed to be at the end of the value. For example, the value 12345 is loaded into a DECIMAL(8,2) column as 123.45, <i>not</i> 12345.00.</p> <p>This modifier cannot be used with the <b>packeddecimal</b> modifier.</p>
<b>timeformat="x"</b>	<p>x is the format of the time in the source file.<sup>1</sup> Valid time elements are:</p> <p>H     - Hour (one or two digits ranging from 0 - 12           for a 12 hour system, and 0 - 24           for a 24 hour system)</p> <p>HH    - Hour (two digits ranging from 00 - 12           for a 12 hour system, and 00 - 24           for a 24 hour system; mutually exclusive           with H)</p> <p>M     - Minute (one or two digits ranging           from 0 - 59)</p> <p>MM    - Minute (two digits ranging from 00 - 59;           mutually exclusive with M)</p> <p>S     - Second (one or two digits ranging           from 0 - 59)</p> <p>SS    - Second (two digits ranging from 00 - 59;           mutually exclusive with S)</p> <p>SSSSS - Second of the day after midnight (5 digits           ranging from 00000 - 86400; mutually           exclusive with other time elements)</p> <p>TT    - Meridian indicator (AM or PM)</p> <p>A default value of 0 is assigned for each element that is not specified. Some examples of time formats are:</p> <p>"HH:MM:SS" "HH.MM TT" "SSSSS"</p>

Table 17. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL) (continued)

Modifier	Description
<b>timestampformat="x"</b>	<p>x is the format of the time stamp in the source file.<sup>1</sup> Valid time stamp elements are:</p> <p>YYYY - Year (four digits ranging from 0000 - 9999)</p> <p>M - Month (one or two digits ranging from 1 - 12)</p> <p>MM - Month (two digits ranging from 01 - 12; mutually exclusive with M and MMM)</p> <p>MMM - Month (three-letter case-insensitive abbreviation for the month name; mutually exclusive with M and MM)</p> <p>D - Day (one or two digits ranging from 1 - 31)</p> <p>DD - Day (two digits ranging from 01 - 31; mutually exclusive with D)</p> <p>DDD - Day of the year (three digits ranging from 001 - 366; mutually exclusive with other day or month elements)</p> <p>H - Hour (one or two digits ranging from 0 - 12 for a 12 hour system, and 0 - 24 for a 24 hour system)</p> <p>HH - Hour (two digits ranging from 00 - 12 for a 12 hour system, and 00 - 24 for a 24 hour system; mutually exclusive with H)</p> <p>M - Minute (one or two digits ranging from 0 - 59)</p> <p>MM - Minute (two digits ranging from 00 - 59; mutually exclusive with M, minute)</p> <p>S - Second (one or two digits ranging from 0 - 59)</p> <p>SS - Second (two digits ranging from 00 - 59; mutually exclusive with S)</p> <p>SSSSS - Second of the day after midnight (5 digits ranging from 00000 - 86400; mutually exclusive with other time elements)</p> <p>U (1 to 12 times)</p> <ul style="list-style-type: none"> <li>- Fractional seconds(number of occurrences of U represent the number of digits with each digit ranging from 0 to 9)</li> </ul> <p>TT - Meridian indicator (AM or PM)</p>
<b>timestampformat="x"</b> (Continued)	<p>A default value of 1 is assigned for unspecified YYYY, M, MM, D, DD, or DDD elements. A default value of 'Jan' is assigned to an unspecified MMM element. A default value of 0 is assigned for all other unspecified elements. The following section is an example of a time stamp format:</p> <p>"YYYY/MM/DD HH:MM:SS.UUUUUU"</p> <p>The valid values for the MMM element include: 'jan', 'feb', 'mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov' and 'dec'. These values are case insensitive.</p> <p>If the <b>timestampformat</b> modifier is not specified, the load utility formats the timestamp field using one of two possible formats:</p> <p>YYYY-MM-DD-HH.MM.SS  YYYY-MM-DD HH:MM:SS</p> <p>The load utility chooses the format by looking at the separator between the DD and HH. If it is a dash '-', the load utility uses the regular dashes and dots format (YYYY-MM-DD-HH.MM.SS). If it is a blank space, then the load utility expects a colon ':' to separate the HH, MM and SS.</p> <p>In either format, if you include the microseconds field (UUUUUU), the load utility expects the dot '.' as the separator. Either YYYY-MM-DD-HH.MM.SS.UUUUUU or YYYY-MM-DD HH:MM:SS.UUUUUU are acceptable.</p> <p>The following example illustrates how to load data containing user defined date and time formats into a table called schedule:</p> <pre>db2 load from delfile2 of del modified by timestampformat="yyyy.mm.dd hh:mm tt" insert into schedule</pre>

Table 17. Valid file type modifiers for the load utility: ASCII file formats (ASC/DEL) (continued)

Modifier	Description
<b>usegraphiccodepage</b>	<p>If <b>usegraphiccodepage</b> is given, the assumption is made that data being loaded into graphic or double-byte character large object (DBCLOB) data field(s) is in the graphic code page. The rest of the data is assumed to be in the character code page. The graphic code page is associated with the character code page. <b>LOAD</b> determines the character code page through either the <b>codepage</b> modifier, if it is specified, or through the code page of the database if the <b>codepage</b> modifier is not specified.</p> <p>This modifier should be used in conjunction with the delimited data file generated by drop table recovery only if the table being recovered has graphic data.</p> <p><b>Restrictions</b></p> <p>The <b>usegraphiccodepage</b> modifier <b>MUST NOT</b> be specified with DEL files created by the <b>EXPORT</b> utility, as these files contain data encoded in only one code page. The <b>usegraphiccodepage</b> modifier is also ignored by the double-byte character large objects (DBCLOBs) in files.</p>
<b>xmlchar</b>	<p>Specifies that XML documents are encoded in the character code page.</p> <p>This option is useful for processing XML documents that are encoded in the specified character code page but do not contain an encoding declaration.</p> <p>For each document, if a declaration tag exists and contains an encoding attribute, the encoding must match the character code page, otherwise the row containing the document will be rejected. The character code page is the value specified by the <b>codepage</b> file type modifier, or the application code page if it is not specified. By default, either the documents are encoded in Unicode, or they contain a declaration tag with an encoding attribute.</p>
<b>xmlgraphic</b>	<p>Specifies that XML documents are encoded in the specified graphic code page.</p> <p>This option is useful for processing XML documents that are encoded in a specific graphic code page but do not contain an encoding declaration.</p> <p>For each document, if a declaration tag exists and contains an encoding attribute, the encoding must match the graphic code page, otherwise the row containing the document will be rejected. The graphic code page is the graphic component of the value specified by the <b>codepage</b> file type modifier, or the graphic component of the application code page if it is not specified. By default, documents are either encoded in Unicode, or they contain a declaration tag with an encoding attribute.</p>

Table 18. Valid file type modifiers for the load utility: ASC file formats (Non-delimited ASCII)

Modifier	Description
<b>binarynumerics</b>	<p>Numeric (but not DECIMAL) data must be in binary form, not the character representation. This avoids costly conversions.</p> <p>This option is supported only with positional ASC, using fixed length records specified by the <b>reclen</b> option.</p> <p>The following rules apply:</p> <ul style="list-style-type: none"> <li>• No conversion between data types is performed, with the exception of BIGINT, INTEGER, and SMALLINT.</li> <li>• Data lengths must match their target column definitions.</li> <li>• FLOATs must be in IEEE Floating Point format.</li> <li>• Binary data in the load source file is assumed to be big-endian, regardless of the platform on which the load operation is running.</li> </ul> <p>NULLs cannot be present in the data for columns affected by this modifier. Blanks (normally interpreted as NULL) are interpreted as a binary value when this modifier is used.</p>
<b>nochecklengths</b>	<p>If <b>nochecklengths</b> is specified, an attempt is made to load each row, even if the source data has a column definition that exceeds the size of the target table column. Such rows can be successfully loaded if code page conversion causes the source data to shrink; for example, 4-byte EUC data in the source could shrink to 2-byte DBCS data in the target, and require half the space. This option is particularly useful if it is known that the source data will fit in all cases despite mismatched column definitions.</p>
<b>nullindchar=<i>x</i></b>	<p><i>x</i> is a single character. Changes the character denoting a NULL value to <i>x</i>. The default value of <i>x</i> is Y.<sup>2</sup></p> <p>This modifier is case sensitive for EBCDIC data files, except when the character is an English letter. For example, if the NULL indicator character is specified to be the letter N, then n is also recognized as a NULL indicator.</p>
<b>packeddecimal</b>	<p>Loads packed-decimal data directly, since the <b>binarynumerics</b> modifier does not include the DECIMAL field type.</p> <p>This option is supported only with positional ASC, using fixed length records specified by the <b>reclen</b> option.</p> <p>Supported values for the sign nibble are:</p> <pre> + = 0xC 0xA 0xE 0xF - = 0xD 0xB </pre> <p>NULLs cannot be present in the data for columns affected by this modifier. Blanks (normally interpreted as NULL) are interpreted as a binary value when this modifier is used.</p> <p>Regardless of the server platform, the byte order of binary data in the load source file is assumed to be big-endian; that is, when using this modifier on Windows operating systems, the byte order must not be reversed.</p> <p>This modifier cannot be used with the <b>implieddecimal</b> modifier.</p>
<b>reclen=<i>x</i></b>	<p><i>x</i> is an integer with a maximum value of 32 767. <i>x</i> characters are read for each row, and a newline character is not used to indicate the end of the row.</p>

Table 18. Valid file type modifiers for the load utility: ASC file formats (Non-delimited ASCII) (continued)

Modifier	Description
<b>striptblanks</b>	<p>Truncates any trailing blank spaces when loading data into a variable-length field. If this option is not specified, blank spaces are kept.</p> <p>This option cannot be specified together with <b>striptnulls</b>. These are mutually exclusive options. This option replaces the obsolete <b>t</b> option, which is supported for earlier compatibility only.</p>
<b>striptnulls</b>	<p>Truncates any trailing NULLs (0x00 characters) when loading data into a variable-length field. If this option is not specified, NULLs are kept.</p> <p>This option cannot be specified together with <b>striptblanks</b>. These are mutually exclusive options. This option replaces the obsolete <b>padwithzero</b> option, which is supported for earlier compatibility only.</p>
<b>zoneddecimal</b>	<p>Loads zoned decimal data, since the <b>binarynumerics</b> modifier does not include the DECIMAL field type. This option is supported only with positional ASC, using fixed length records specified by the <b>recLEN</b> option.</p> <p>Half-byte sign values can be one of the following value:            + = 0xC 0xA 0xE 0xF 0x3            - = 0xD 0xB 0x7</p> <p>Supported values for digits are 0x0 to 0x9.</p> <p>Supported values for zones are 0x3 and 0xF.</p>

Table 19. Valid file type modifiers for the load utility: DEL file formats (Delimited ASCII)

Modifier	Description
<b>charde1x</b>	<p><i>x</i> is a single character string delimiter. The default value is a double quotation mark ("). The specified character is used in place of double quotation marks to enclose a character string.<sup>2, 3</sup> If you want to explicitly specify the double quotation mark (") as the character string delimiter, you should specify it as follows:            modified by charde1""</p> <p>The single quotation mark (') can also be specified as a character string delimiter as follows:            modified by charde1''</p>
<b>colde1x</b>	<p><i>x</i> is a single character column delimiter. The default value is a comma (,). The specified character is used in place of a comma to signal the end of a column.<sup>2,3</sup></p>
<b>decplusblank</b>	<p>Plus sign character. Causes positive decimal values to be prefixed with a blank space instead of a plus sign (+). The default action is to prefix positive decimal values with a plus sign.</p>
<b>decptx</b>	<p><i>x</i> is a single character substitute for the period as a decimal point character. The default value is a period (.). The specified character is used in place of a period as a decimal point character.<sup>2, 3</sup></p>

Table 19. Valid file type modifiers for the load utility: DEL file formats (Delimited ASCII) (continued)

Modifier	Description
<b>delprioritychar</b>	<p>The current default priority for delimiters is: record delimiter, character delimiter, column delimiter. This modifier protects existing applications that depend on the older priority by reverting the delimiter priorities to: character delimiter, record delimiter, column delimiter. Syntax:</p> <pre>db2 load ... modified by delprioritychar ...</pre> <p>For example, given the following DEL data file:</p> <pre>"Smith, Joshua",4000,34.98&lt;row delimiter&gt; "Vincent,&lt;row delimiter&gt;, is a manager", ... ... 4005,44.37&lt;row delimiter&gt;</pre> <p>With the <b>delprioritychar</b> modifier specified, there will be only two rows in this data file. The second &lt;row delimiter&gt; will be interpreted as part of the first data column of the second row, while the first and the third &lt;row delimiter&gt; are interpreted as actual record delimiters. If this modifier is <i>not</i> specified, there will be three rows in this data file, each delimited by a &lt;row delimiter&gt;.</p>
<b>keepblanks</b>	<p>Preserves the leading and trailing blanks in each field of type CHAR, VARCHAR, LONG VARCHAR, or CLOB. Without this option, all leading and trailing blanks that are not inside character delimiters are removed, and a NULL is inserted into the table for all blank fields.</p> <p>The following example illustrates how to load data into a table called TABLE1, while preserving all leading and trailing spaces in the data file:</p> <pre>db2 load from delfile3 of del     modified by keepblanks     insert into table1</pre>
<b>nocharde1</b>	<p>The load utility will assume all bytes found between the column delimiters to be part of the column's data. Character delimiters will be parsed as part of column data. This option should not be specified if the data was exported using a Db2 database system (unless <b>nocharde1</b> was specified at export time). It is provided to support vendor data files that do not have character delimiters. Improper usage might result in data loss or corruption.</p> <p>This option cannot be specified with <b>charde1x</b>, <b>delprioritychar</b> or <b>nodoublede1</b>. These are mutually exclusive options.</p>
<b>nodoublede1</b>	Suppresses recognition of double character delimiters.

Table 20. Valid file type modifiers for the load utility: IXF file format

Modifier	Description
<b>forcein</b>	<p>Directs the utility to accept data despite code page mismatches, and to suppress translation between code pages.</p> <p>Fixed length target fields are checked to verify that they are large enough for the data. If <b>nochecklengths</b> is specified, no checking is done, and an attempt is made to load each row.</p>
<b>nochecklengths</b>	If <b>nochecklengths</b> is specified, an attempt is made to load each row, even if the source data has a column definition that exceeds the size of the target table column. Such rows can be successfully loaded if code page conversion causes the source data to shrink; for example, 4-byte EUC data in the source could shrink to 2-byte DBCS data in the target, and require half the space. This option is particularly useful if it is known that the source data will fit in all cases despite mismatched column definitions.

## Notes

1. Double quotation marks around the date format string are mandatory. Field separators cannot contain any of the following characters: a-z, A-Z, and 0-9. The field separator should not be the same as the character delimiter or field delimiter in the DEL file format. A field separator is optional if the start and end positions of an element are unambiguous. Ambiguity can exist if (depending on the modifier) elements such as D, H, M, or S are used, because of the variable length of the entries.

For time stamp formats, care must be taken to avoid ambiguity between the month and the minute descriptors, since they both use the letter M. A month field must be adjacent to other date fields. A minute field must be adjacent to other time fields. Following are some ambiguous time stamp formats:

```
"M" (could be a month, or a minute)
"M:M" (Which is which?)
"M:YYYY:M" (Both are interpreted as month.)
"S:M:YYYY" (adjacent to both a time value and a date value)
```

In ambiguous cases, the utility will report an error message, and the operation will fail.

Following are some unambiguous time stamp formats:

```
"M:YYYY" (Month)
"S:M" (Minute)
"M:YYYY:S:M" (Month....Minute)
"M:H:YYYY:M:D" (Minute....Month)
```

Some characters, such as double quotation marks and backslashes, must be preceded by an escape character (for example, \).

2. Character values provided for the **chardef1**, **coldef1**, or **decpt** file type modifiers must be specified in the code page of the source data.

The character code point (instead of the character symbol), can be specified using the syntax xJJ or 0xJJ, where JJ is the hexadecimal representation of the code point. For example, to specify the # character as a column delimiter, use one of the following statements:

```
... modified by coldef1# ...
... modified by coldef10x23 ...
... modified by coldef1X23 ...
```

3. "Delimiter considerations for moving data" lists restrictions that apply to the characters that can be used as delimiter overrides.
4. The load utility does not issue a warning if an attempt is made to use unsupported file types with the **MODIFIED BY** option. If this is attempted, the load operation fails, and an error code is returned.
5. When multiple modifiers suffixed with **ignore**, **include**, **missing**, and **override** are specified, they are applied in the order that they are listed. In the following statement, data for implicitly hidden columns that are not identity columns is included in the input data. While data for all identity columns, regardless of their implicitly hidden status, is not.

```
db2 load from delfile1 of del modified by
 implicitlyhiddeninclude identitymissing insert into table1
```

However, changing the order of the file type modifiers in the following statement means that data for all implicitly hidden columns (including hidden identity columns) is included in the input data. While data for identity columns that are not implicitly hidden is not.

```
db2 load from delfile1 of del modified by
 identitymissing implicitlyhiddeninclude insert into table1
```

Table 21. LOAD behavior when using code page and usegraphiccodepage

codepage=N	usegraphiccodepage	LOAD behavior
Absent	Absent	All data in the file is assumed to be in the database code page, not the application code page, even if the <b>CLIENT</b> option is specified.
Present	Absent	All data in the file is assumed to be in code page N.  <b>Warning:</b> Graphic data will be corrupted when loaded into the database if N is a single-byte code page.
Absent	Present	Character data in the file is assumed to be in the database code page, even if the <b>CLIENT</b> option is specified. Graphic data is assumed to be in the code page of the database graphic data, even if the <b>CLIENT</b> option is specified.  If the database code page is single-byte, then all data is assumed to be in the database code page.  <b>Warning:</b> Graphic data will be corrupted when loaded into a single-byte database.
Present	Present	Character data is assumed to be in code page N. Graphic data is assumed to be in the graphic code page of N.  If N is a single-byte or double-byte code page, then all data is assumed to be in code page N.  <b>Warning:</b> Graphic data will be corrupted when loaded into the database if N is a single-byte code page.

#### PRUNE HISTORY/LOGFILE command using the ADMIN\_CMD procedure:

Used to delete entries from the recovery history file or to delete log files from the active log file path. Deleting entries from the recovery history file might be necessary if the file becomes excessively large and the retention period is high.

In a partitioned environment, the **PRUNE HISTORY** command only performs on the database partition it is issued on. To prune the history on multiple partitions, you can either issue the **PRUNE HISTORY** command from each individual database partition, or use the db2\_all prefix to run the **PRUNE HISTORY** command on all database partitions.

**Important:** The **PRUNE LOGFILE** command is deprecated and might be removed in a future release. Use the **PRUNE HISTORY** command instead.

#### Authorization

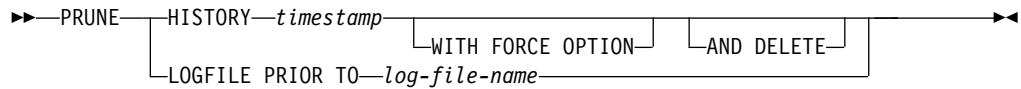
One of the following authorities:

- SYSADM
- SYSCtrl
- SYSMAINT
- DBADM

#### Required connection

Database

## Command syntax



## Command parameters

### **HISTORY *timestamp***

Identifies a range of entries in the recovery history file that will be deleted. A complete time stamp (in the form *yyyymmddhhmmss*), or an initial prefix (minimum *yyyy*) can be specified. All entries with time stamps equal to or less than the time stamp provided are deleted from the recovery history file. When an initial prefix is specified, the unspecified components of the time stamp are interpreted as *yyyy0101000000*.

### **WITH FORCE OPTION**

Specifies that the entries will be pruned according to the time stamp specified, even if some entries from the most recent restore set are deleted from the file. A restore set is the most recent full database backup including any restores of that backup image. If this parameter is not specified, all entries from the backup image forward will be maintained in the history.

### **AND DELETE**

Specifies that the associated log archives will be physically deleted (based on the location information) when the history file entry is removed. This option is especially useful for ensuring that archive storage space is recovered when log archives are no longer needed. If you are archiving logs via a user exit program, the logs cannot be deleted using this option.

If you set the **auto\_del\_rec\_obj** database configuration parameter to ON, calling **PRUNE HISTORY** with the **AND DELETE** parameter will also physically delete backup images and load copy images if their history file entry is pruned.

### **LOGFILE PRIOR TO *log-file-name***

Specifies a string for a log file name, for example *S0000100.LOG*. All log files before (but not including) the specified log file will be deleted. The **logarchmeth1** database configuration parameter must be set to a value other than **OFF**.

**Note:** This value is not supported in Db2 pureScale environments.

## Examples

To remove the entries for all restores, loads, table space backups, and full database backups taken before and including December 1, 1994 from the recovery history file, enter:

```
db2 prune history 199412
```

199412 is interpreted as 19941201000000.

## Usage notes

If the **WITH FORCE OPTION** is used, you might delete entries that are required for automatic restoration of databases. Manual restores will still work correctly. Use of

this command can also prevent the **db2ckrst** utility from being able to correctly analyze the complete chain of required backup images. Using the **PRUNE HISTORY** command without the **WITH FORCE OPTION** prevents required entries from being deleted.

Those entries with status `DB2HISTORY_STATUS_DO_NOT_DELETE` will not be pruned. If the **WITH FORCE OPTION** is used, then objects marked as `DB2HISTORY_STATUS_DO_NOT_DELETE` will still be pruned or deleted. You can set the status of recovery history file entries to `DB2HISTORY_STATUS_DO_NOT_DELETE` using the **UPDATE HISTORY** command, the `ADMIN_CMD` with **UPDATE\_HISTORY**, or the `db2HistoryUpdate` API. You can use the `DB2HISTORY_STATUS_DO_NOT_DELETE` status to prevent key recovery history file entries from being pruned and to prevent associated recovery objects from being deleted.

You can prune snapshot backup database history file entries using the **PRUNE HISTORY** command, but you cannot delete the related physical recovery objects using the **AND DELETE** parameter. The only way to delete snapshot backup object is to use the **db2acsutil** command.

#### **QUIESCE DATABASE command using the ADMIN\_CMD procedure:**

Forces all users off either the specified instance or database across all members and puts them into a quiesced mode.

While the instance or database is in quiesced mode, you can perform administrative tasks on it. After administrative tasks are complete, use the **UNQUIESCE** command to activate the instance or database and allow other users to connect to the database.

In this mode, only users with authority in this restricted mode are allowed to attach or connect to the instance or database. Users with `SYSADM`, `SYSMAINT`, and `SYSCTRL` authority always have access to an instance while it is quiesced, and users with `SYSADM` and `DBADM` authority always have access to a database while it is quiesced.

#### **Scope**

**QUIESCE DATABASE** results in all objects in the database being in the quiesced mode. Only the allowed user or group and `SYSADM`, `SYSMAINT`, `DBADM`, or `SYSCTRL` will be able to access the database or its objects.

**QUIESCE INSTANCE** *instance-name* means the instance and the databases in the instance *instance-name* will be in quiesced mode. The instance will be accessible just for `SYSADM`, `SYSMAINT`, and `SYSCTRL` and allowed user or group.

If an instance is in quiesced mode, a database in the instance cannot be put in quiesced mode.

If a database is in the `SUSPEND_WRITE` state, it cannot be put in quiesced mode.

#### **Authorization**

One of the following authorities:

For database level quiesce:

- SYSADM
- DBADM

For instance level quiesce:

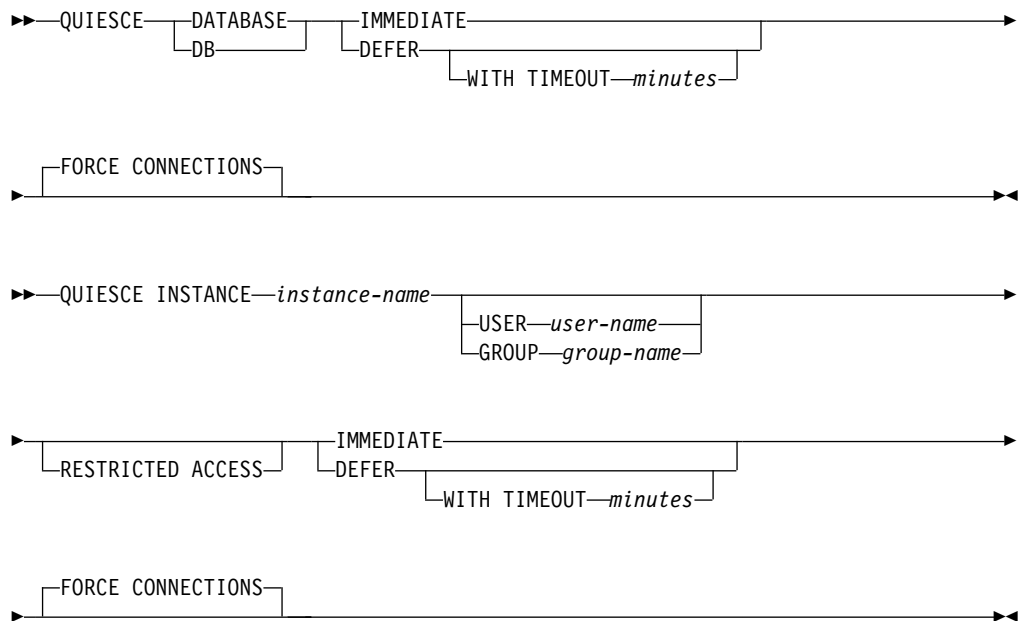
- SYSADM
- SYSCtrl

### Required connection

Database

(Database connection is not required for an instance quiesce.)

### Command syntax



### Command parameters

**DEFER** Wait for applications until they commit the current unit of work.

#### **WITH TIMEOUT *minutes***

Specifies a time, in minutes, to wait for applications to commit the current unit of work. If no value is specified, in a single-partition database environment, the default value is 10 minutes. In a partitioned database environment the value specified by the **start\_stop\_time** database manager configuration parameter will be used.

#### **IMMEDIATE**

Do not wait for the transactions to be committed, immediately roll back the transactions.

#### **FORCE CONNECTIONS**

Force the connections off.

#### **DATABASE**

Quiesce the database. All objects in the database will be placed in quiesced

mode. Only specified users in specified groups and users with SYSADM, SYSMAINT, and SYSCTRL authority will be able to access to the database or its objects.

**INSTANCE** *instance-name*

The instance *instance-name* and the databases in the instance will be placed in quiesced mode. The instance will be accessible only to users with SYSADM, SYSMAINT, and SYSCTRL authority and specified users in specified groups.

The databases within the quiesced instance will be activated to do authorization checking for all connect attempts to the database. This is necessary to determine if the connecting user ID has DBADM authority; this authority is stored in the database catalog and the database must be activated to determine if the user ID has it. To prevent this authorization checking from happening, specify the **RESTRICTED ACCESS** option.

**USER** *user-name*

Specifies the name of a user who will be allowed access to the instance while it is quiesced.

**GROUP** *group-name*

Specifies the name of a group that will be allowed access to the instance while the instance is quiesced.

**RESTRICTED ACCESS**

Specify this option to prevent authorization checking for all connect attempts to the databases of a quiesced instance to determine if the user ID has DBADM authority. Instance-level authorization checking can still occur; checking a user ID for SYSADM, SYSCTRL, or SYSMAINT authority does not require a database to be activated.

**Note:** This command option is available in Db2 Version 9.7 Fix Pack 2 and later fix packs.

**Examples**

In the following example, the default behavior is to force connections, so it does not need to be explicitly stated and can be removed from this example.

```
db2 quiesce instance crankarm user frank immediate force connections
```

The following example forces off all users with connections to the database.

```
db2 quiesce db immediate
```

- The first example will quiesce the instance crankarm, while allowing user frank to continue using the database.

The second example will quiesce the database you are attached to, preventing access by all users except those with one of the following authorities: SYSADM, SYSMAINT, SYSCTRL, or DBADM.

- This command will force all users off the database or instance if **FORCE CONNECTIONS** option is supplied. **FORCE CONNECTIONS** is the default behavior; the parameter is allowed in the command for compatibility reasons.
- The command will be synchronized with the **FORCE CONNECTIONS** and will only complete once the **FORCE CONNECTIONS** has completed.

### Usage notes

- After **QUIESCE INSTANCE**, only users with SYSADM, SYSMAINT, or SYSCTRL authority or a user name and group name provided as parameters to the command can connect to the instance.
- The **RESTRICTED ACCESS** option will prevent the databases within the quiesced instance from being activated to do authorization checking. Any user ID trying to connect to a database, which has DBADM authority or QUIESCE\_CONNECT privilege on the database, will not be allowed to connect. Only user IDs which have SYSADM, SYSCTRL, or SYSMAINT authority and the user or group specified with the command will be allowed to connect to the database.
- The **RESTRICTED ACCESS** option should be used when there is a need to have exclusive connections to a database within the quiesced instance. Such cases can include making an offline backup or performing other maintenance activities.
- After **QUIESCE DATABASE**, users with SYSADM, SYSMAINT, SYSCTRL, or DBADM authority, and GRANT or REVOKE privileges can designate who will be able to connect. This information will be stored permanently in the database catalog tables.

For example,

```
grant quiesce_connect on database to username/groupname
revoke quiesce_connect on database from username/groupname
```

- In a Db2 pureScale environment, after quiescing a database and restarting the instance, the database will remain quiesced across all members. An explicit **UNQUIESCE DATABASE** command is required to remove the quiesce state.

### QUIESCE TABLESPACES FOR TABLE command using the ADMIN\_CMD procedure:

Quiesces table spaces for a table. There are three valid quiesce modes: share, intent to update, and exclusive.

There are three possible states resulting from the quiesce function:

- Quiesced: SHARE
- Quiesced: UPDATE
- Quiesced: EXCLUSIVE

### Scope

In a single-partition environment, this command quiesces all table spaces involved in a load operation in exclusive mode for the duration of the load operation. In a partitioned database environment, this command acts locally on a database partition. It quiesces only that portion of table spaces belonging to the database partition on which the load operation is performed. For partitioned tables, all of the table spaces listed in SYSDATAPARTITIONS.TBSPACEID and SYSDATAPARTITIONS.LONG\_TBSPACEID associated with a table and with a status of normal, attached or detached, (for example, SYSDATAPARTITIONS.STATUS of '', 'A' or 'D') are quiesced.

### Authorization

One of the following authorities:

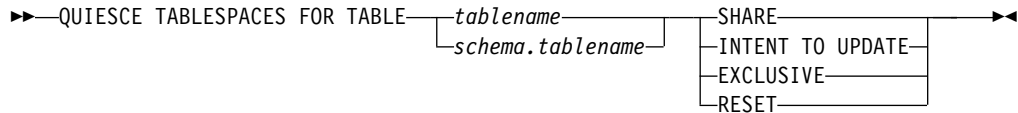
- SYSADM
- SYSCTRL
- SYSMAINT

- DBADM
- LOAD

## Required connection

Database

## Command syntax



## Command parameters

### TABLE

#### *tablename*

Specifies the unqualified table name. The table cannot be a system catalog table.

#### *schema.tablename*

Specifies the qualified table name. If *schema* is not provided, the CURRENT SCHEMA will be used. The table cannot be a system catalog table.

**SHARE** Specifies that the quiesce is to be in share mode.

When a "quiesce share" request is made, the transaction requests intent share locks for the table spaces and a share lock for the table. When the transaction obtains the locks, the state of the table spaces is changed to QUIESCED SHARE. The state is granted to the quiescer only if there is no conflicting state held by other users. The state of the table spaces, along with the authorization ID and the database agent ID of the quiescer, are recorded in the table space table, so that the state is persistent. The table cannot be changed while the table spaces for the table are in QUIESCED SHARE state. Other share mode requests to the table and table spaces are allowed. When the transaction commits or rolls back, the locks are released, but the table spaces for the table remain in QUIESCED SHARE state until the state is explicitly reset.

### INTENT TO UPDATE

Specifies that the quiesce is to be in intent to update mode.

When a "quiesce intent to update" request is made, the table spaces are locked in intent exclusive (IX) mode, and the table is locked in update (U) mode. The state of the table spaces is recorded in the table space table.

### EXCLUSIVE

Specifies that the quiesce is to be in exclusive mode.

When a "quiesce exclusive" request is made, the transaction requests super exclusive locks on the table spaces, and a super exclusive lock on the table. When the transaction obtains the locks, the state of the table spaces changes to QUIESCED EXCLUSIVE. The state of the table spaces, along with the authorization ID and the database agent ID of the quiescer, are recorded in the table space table. Since the table spaces are held in super

exclusive mode, no other access to the table spaces is allowed. The user who invokes the quiesce function (the quiescer) has exclusive access to the table and the table spaces.

**RESET** Specifies that the state of the table spaces is to be reset to normal. A quiesce state cannot be reset if the connection that issued the quiesce request is still active.

When a quiescer issues a reset, only the quiesce mode for that quiescer is reset. If there are multiple quiescers, then the state of the table space will appear unchanged.

When working with a system-period temporal table and its associated history table, the reset operation must be performed on the same table that was used to originally set the quiesce mode.

### Example

```
db2 quiesce tablespaces for table staff share
db2 quiesce tablespaces for table boss.org intent to update
```

### Usage notes

This command is not supported in Db2 pureScale environments.

A quiesce is a persistent lock. Its benefit is that it persists across transaction failures, connection failures, and even across system failures (such as power failure, or reboot).

A quiesce is owned by a connection. If the connection is lost, the quiesce remains, but it has no owner, and is called a *phantom quiesce*. For example, if a power outage caused a load operation to be interrupted during the delete phase, the table spaces for the loaded table would be left in quiesce exclusive state. Upon database restart, this quiesce would be an unowned (or phantom) quiesce. The removal of a phantom quiesce requires a connection with the same user ID used when the quiesce mode was set.

To remove a phantom quiesce:

1. Connect to the database with the same user ID used when the quiesce mode was set.
2. Use the **LIST TABLESPACES** command to determine which table space is quiesced.
3. Re-quiesce the table space using the current quiesce state. For example:  

```
db2 quiesce tablespaces for table mytable exclusive
```

Once completed, the new connection owns the quiesce, and the load operation can be restarted.

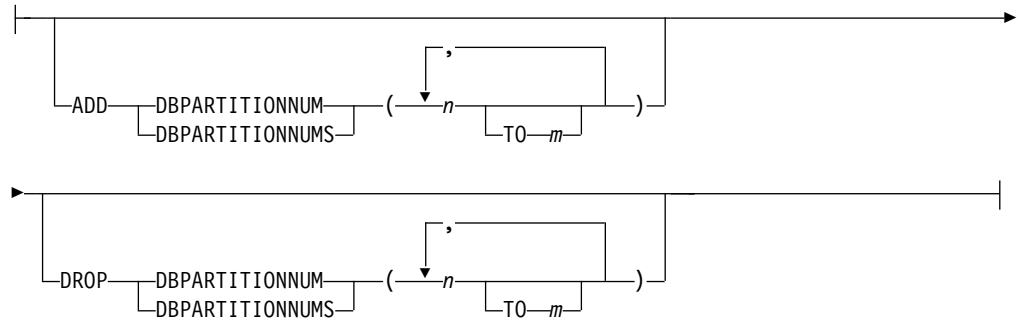
There is a limit of five quiescers on a table space at any given time.

A quiescer can alter the state of a table space from a less restrictive state to a more restrictive one (for example, S to U, or U to X). If a user requests a state lower than one that is already held, the original state is returned. States are not downgraded.

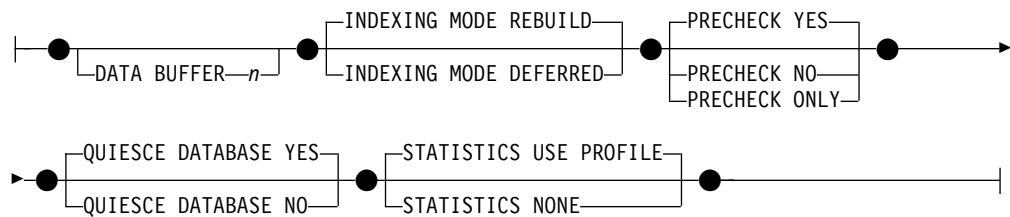
When quiescing against a system-period temporal table, all the table spaces associated with the system-period temporal table and the history table are quiesced. When quiescing against a history table, all the table spaces associated



### Add/Drop DB partition:



### Not rollforward recoverable options:



### Command parameters

#### **DATABASE PARTITION GROUP** *db-partition-group*

The name of the database partition group. This one-part name identifies a database partition group described in the SYSCAT.DBPARTITIONGROUPS catalog table. The database partition group cannot currently be undergoing redistribution.

**Note:** Tables in the IBMCATGROUP and the IBMTEMPGROUP database partition groups cannot be redistributed.

#### **NOT ROLLFORWARD RECOVERABLE**

When this option is used, the **REDISTRIBUTE DATABASE PARTITION GROUP** command is not rollforward recoverable.

- Data is moved in bulk instead of by internal insert and delete operations. This reduces the number of times that a table must be scanned and accessed, which results in better performance.
- Log records are no longer required for each of the insert and delete operations. This means that you no longer need to manage large amounts of active log space and log archiving space in your system when performing data redistribution.
- When using the **REDISTRIBUTE DATABASE PARTITION GROUP** command with the **NOT ROLLFORWARD RECOVERABLE** option, the redistribute operation uses the **INDEXING MODE DEFERRED** option for tables that contain XML columns. If a table does not contain an XML column, the redistribute operation uses the indexing mode specified when issuing the command.

When this option is *not* used, extensive logging of all row movement is performed such that the database can be recovered later in the event of any interruptions, errors, or other business need.

This option is not supported for column-organized tables.

## UNIFORM

Specifies that the data is uniformly distributed across hash partitions (that is, every hash partition is assumed to have the same number of rows), but the same number of hash partitions do not map to each database partition. After redistribution, all database partitions in the database partition group have approximately the same number of hash partitions.

## USING DISTFILE *distfilename*

If the distribution of distribution key values is skewed, use this option to achieve a uniform redistribution of data across the database partitions of a database partition group.

Use the *distfilename* to indicate the current distribution of data across the 32 768 hash partitions.

Use row counts, byte volumes, or any other measure to indicate the amount of data represented by each hash partition. The utility reads the integer value associated with a partition as the weight of that partition. When a *distfilename* is specified, the utility generates a target distribution map that it uses to redistribute the data across the database partitions in the database partition group as uniformly as possible. After the redistribution, the weight of each database partition in the database partition group is approximately the same (the weight of a database partition is the sum of the weights of all hash partitions that map to that database partition).

For example, the input distribution file might contain entries as follows:

```
10223
1345
112000
0
100
...
```

In the example, hash partition 2 has a weight of 112000, and partition 3 (with a weight of 0) has no data mapping to it at all.

The *distfilename* should contain 32 768 positive integer values in character format. The sum of the values should be less than or equal to 4 294 967 295.

If the path for *distfilename* is not specified, the current directory is used.

## USING TARGETMAP *targetmapfilename*

The file specified in *targetmapfilename* is used as the target distribution map. Data redistribution is done according to this file. If the path is not specified, the current directory is used.

The *targetmapfilename* should contain 32 768 integers, each representing a valid database partition number. The number on any row maps a hash value to a database partition. This means that if row *X* contains value *Y*, then every record with HASHEDVALUE() of *X* is to be located on database partition *Y*.

If a database partition, included in the target map, is not in the database partition group, an error is returned. Issue ALTER DATABASE PARTITION GROUP ADD DBPARTITIONNUM statement before running **REDISTRIBUTE DATABASE PARTITION GROUP** command.

If a database partition, excluded from the target map, is in the database partition group, that database partition will not be included in the partitioning. Such a database partition can be dropped using ALTER DATABASE PARTITION GROUP DROP DBPARTITIONNUM statement either before or after the **REDISTRIBUTE DATABASE PARTITION GROUP** command.

**CONTINUE**

Continues a previously failed or stopped **REDISTRIBUTE DATABASE PARTITION GROUP** operation. If none occurred, an error is returned.

**ABORT**

Aborts a previously failed or stopped **REDISTRIBUTE DATABASE PARTITION GROUP** operation. If none occurred, an error is returned.

**ADD**

**DBPARTITIONNUM *n***

**TO *m***

*n* or *n TO m* specifies a list or lists of database partition numbers which are to be added into the database partition group. Any specified partition must not already be defined in the database partition group (SQLSTATE 42728). This is equivalent to executing the ALTER DATABASE PARTITION GROUP statement with ADD DBPARTITIONNUM clause specified.

**DBPARTITIONNUMS *n***

**TO *m***

*n* or *n TO m* specifies a list or lists of database partition numbers which are to be added into the database partition group. Any specified partition must not already be defined in the database partition group (SQLSTATE 42728). This is equivalent to executing the ALTER DATABASE PARTITION GROUP statement with ADD DBPARTITIONNUM clause specified.

**Note:**

1. When a database partition is added using this option, containers for table spaces are based on the containers of the corresponding table space on the lowest numbered existing partition in the database partition group. If this would result in a naming conflict among containers, which could happen if the new partitions are on the same physical machine as existing containers, this option should not be used. Instead, the ALTER DATABASE PARTITION GROUP statement should be used with the WITHOUT TABLESPACES option before issuing the **REDISTRIBUTE DATABASE PARTITION GROUP** command. Table space containers can then be created manually specifying appropriate names.
2. Data redistribution might create table spaces for all new database partitions if the **ADD DBPARTITIONNUMS** parameter is specified.

**DROP**

**DBPARTITIONNUM *n***

**TO *m***

*n* or *n TO m* specifies a list or lists of database partition numbers which are to be dropped from the database partition group. Any specified partition must already be defined in the database partition group (SQLSTATE 42729). This is equivalent to executing the ALTER DATABASE PARTITION GROUP statement with the DROP DBPARTITIONNUM clause specified.

**DBPARTITIONNUMS *n***

**TO *m***

*n* or *n TO m* specifies a list or lists of database partition numbers which are to be dropped from the database partition group. Any specified partition must already be defined in the database partition group (SQLSTATE 42729). This is equivalent to executing the ALTER DATABASE PARTITION GROUP statement with the DROP DBPARTITIONNUM clause specified.

**TABLE *tablename***

Specifies a table order for redistribution processing.

**ONLY**

If the table order is followed by the **ONLY** keyword (which is the default), then, only the specified tables will be redistributed. The remaining tables can be later processed by **REDISTRIBUTE CONTINUE** commands. This is the default.

**FIRST**

If the table order is followed by the **FIRST** keyword, then, the specified tables will be redistributed with the given order and the remaining tables in the database partition group will be redistributed with random order.

**EXCLUDE *tablename***

Specifies tables to omit from redistribution processing. For example, you can temporarily omit a table until you can configure it to meet the requirements for data redistribution. The omitted tables can be later processed by **REDISTRIBUTE CONTINUE** commands.

**STOP AT *local-isotime***

When this option is specified, before beginning data redistribution for each table, the *local-isotime* is compared with the current local timestamp. If the specified *local-isotime* is equal to or earlier than the current local timestamp, the utility stops with a warning message. Data redistribution processing of tables in progress at the stop time will complete without interruption. No new data redistribution processing of tables begins. The unprocessed tables can be redistributed using the **CONTINUE** option. This *local-isotime* value is specified as a time stamp, a 7-part character string that identifies a combined date and time. The format is *yyyy-mm-dd-hh.mm.ss.nnnnnn* (year, month, day, hour, minutes, seconds, microseconds) expressed in local time.

**DATA BUFFER *n***

Specifies the number of 4 KB pages to use as buffered space for transferring data within the utility. This command parameter can be used only when the **NOT ROLLFORWARD RECOVERABLE** parameter is also specified.

If the value specified is lower than the minimum supported value, the minimum value is used and no warning is returned. If a **DATA BUFFER** value is not specified, an intelligent default is calculated by the utility at runtime at the beginning of processing each table. Specifically, the default is to use 50% of the memory available in the utility heap at the time redistribution of the table begins and to take into account various table properties as well.

This memory is allocated directly from the utility heap, whose size can be modified through the **util\_heap\_sz** database configuration parameter. The value of the **DATA BUFFER** parameter of the **REDISTRIBUTE DATABASE PARTITION GROUP** command can temporarily exceed **util\_heap\_sz** if more memory is available in the system.

**INDEXING MODE**

Specifies how indexes are maintained during redistribution. This command parameter can be used only when the **NOT ROLLFORWARD RECOVERABLE** parameter is also specified.

Valid values are:

#### **REBUILD**

Indexes will be rebuilt from scratch. Indexes do not have to be valid to use this option. As a result of using this option, index pages will be clustered together on disk.

#### **DEFERRED**

Redistribute will not attempt to maintain any indexes. Indexes will be marked as needing a refresh. The first access to such indexes might force a rebuild, or indexes might be rebuilt when the database is restarted.

**Note:** For non-MDC and non-ITC tables, if there are invalid indexes on the tables, the **REDISTRIBUTE DATABASE PARTITION GROUP** command automatically rebuilds them if you do not specify **INDEXING MODE DEFERRED**. For an MDC or ITC table, even if you specify **INDEXING MODE DEFERRED**, a composite index that is invalid is rebuilt before table redistribution begins because the utility needs the composite index to process an MDC or ITC table.

#### **PRECHECK**

Verifies that the database partition group can be redistributed. This command parameter can be used only when the **NOT ROLLFORWARD RECOVERABLE** parameter is also specified.

##### **YES**

This is the default value. The redistribution operation begins only if the verification completes successfully. If the verification fails, the command terminates and returns an error message related to the first check that failed.

**NO** The redistribution operation begins immediately; no verification occurs.

##### **ONLY**

The command terminates after performing the verification; no redistribution occurs. By default it will not quiesce the database. If the **QUIESCE DATABASE** command parameter was set to YES or defaulted to a value of YES, the database remains quiesced. To restore connectivity to the database, perform the redistribution operation or issue **UNQUIESCE DATABASE** command.

#### **QUIESCE DATABASE**

Specifies to force all users off the database and put it into a quiesced mode. This command parameter can be used only when the **NOT ROLLFORWARD RECOVERABLE** parameter is also specified.

##### **YES**

This is the default value. Only users with SYSADM, SYSMAINT, or SYSCTRL authority or users who have been granted QUIESCE\_CONNECT authority will be able to access the database or its objects. Once the redistribution completes successfully, the database is unquiesced.

**NO** The redistribution operation does not quiesce the database; no users are forced off the database.

For more information, refer to the **QUIESCE DATABASE** command.

#### **STATISTICS**

Specifies that the utility should collect statistics for the tables that have a statistics profile. This command parameter can be used only when the **NOT ROLLFORWARD RECOVERABLE** parameter is also specified.

Specifying this option is more efficient than separately issuing the **RUNSTATS** command after the data redistribution is completed.

#### USE PROFILE

Statistics will be collected for the tables with a statistics profile. For tables without a statistics profile, nothing will be done. This is the default.

#### NONE

Statistics will not be collected for tables.

### Examples

Redistribute database partition group DBPG\_1 by providing the current data distribution through a data distribution file, `distfile_for_dbpg_1`. Move the data onto two new database partitions, 6 and 7.

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1
 USING DISTFILE /home/user1/data/distfile_for_dbpg_1
 ADD DATABASE PARTITION (6 TO 7)
```

Redistribute database partition group DBPG\_2 such that:

- The redistribution is not rollforward recoverable;
- Data is uniformly distributed across hash partitions;
- Indexes are rebuilt from scratch;
- Statistics are not collected;
- 180,000 4 KB pages are used as buffered space for transferring the data.

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_2
 NOT ROLLFORWARD RECOVERABLE
 UNIFORM
 INDEXING MODE REBUILD
 DATA BUFFER 180000
 STATISTICS NONE
```

This redistribution operation also quiesces the database and performs a precheck due to the default values for the **QUIESCE DATABASE** and **PRECHECK** command parameters.

### Usage notes

- Before starting a redistribute operation, ensure that the tables are in normal state and not in "load pending" state or "reorg pending" state. Table states can be checked by using the **LOAD QUERY** command.
- When the **NOT ROLLFORWARD RECOVERABLE** option is specified and the database is a recoverable database, the first time the utility accesses a table space, it is put into the BACKUP PENDING state. All the tables in that table space will become read-only until the table space is backed-up, which can only be done when all tables in the table space have finished being redistributed.
- When a redistribution operation is running, it produces an event log file containing general information about the redistribution operation and information such as the starting and ending time of each table processed. This event log file is written to:
  - The `homeinst/sqllib/redist` directory on Linux and UNIX operating systems, using the following format for subdirectories and file name:  
*database-name.database-partition-group-name.timestamp.log*.
  - The `DB2INSTPROF\instance\redist` directory on Windows operating systems (where **DB2INSTPROF** is the value of the **DB2INSTPROF** registry variable), using

the following format for subdirectories and file name: *database-name.database-partition-group-name.timestamp.log*.

- The time stamp value is the time when the command was issued.
- This utility performs intermittent COMMITs during processing.
- All packages having a dependency on a table that has undergone redistribution are invalidated. It is recommended to explicitly rebind such packages after the redistribute database partition group operation has completed. Explicit rebinding eliminates the initial delay in the execution of the first SQL request for the invalid package. The redistribute message file contains a list of all the tables that have undergone redistribution.
- By default, the redistribute utility will update the statistics for those tables that have a statistics profile. For the tables without a statistics profile, it is recommended that you separately update the table and index statistics for these tables by calling the db2Runstats API or by issuing the **RUNSTATS** command after the redistribute operation has completed.
- Database partition groups containing replicated materialized query tables or tables defined with DATA CAPTURE CHANGES cannot be redistributed.
- Redistribution is not allowed if there are user temporary table spaces with existing declared temporary tables or created temporary tables in the database partition group.
- Options such as **INDEXING MODE** are ignored on tables, on which they do not apply, without warning. For example, **INDEXING MODE** will be ignored on tables without indexes.
- The **REDISTRIBUTE DATABASE PARTITION GROUP** command might fail (SQLSTATE 55071) if an add database partition server request is either pending or in progress. This command might also fail (SQLSTATE 55077) if a new database partition server is added online to the instance and not all applications are aware of the new database partition server.
- The **REDISTRIBUTE DATABASE PARTITION GROUP** command is not allowed if there is an outstanding asynchronous background process to create a column compression dictionary (SQL6056N).

## Compatibilities

Tables containing XML columns that use the Db2 Version 9.5 or earlier XML record format cannot be redistributed. Use the **ADMIN\_MOVE\_TABLE** stored procedure to migrate the table to the new format.

## REORG INDEXES/TABLE command using the ADMIN\_CMD procedure:

Reorganizes an index or a table.

You can reorganize all indexes that are defined on a table by rebuilding the index data into unfragmented, physically contiguous pages. On a data partitioned table, you can reorganize a specific nonpartitioned index on a partitioned table, or you can reorganize all the partitioned indexes on a specific data partition.

If you specify the **CLEANUP** option of the index clause, cleanup is performed without rebuilding the indexes. You can cleanup a specific index on a table, or you can cleanup all of the indexes on the table.

This command cannot be used against indexes on declared temporary tables or created temporary tables (SQLSTATE 42995).

The table option reorganizes a table by reconstructing the rows to eliminate fragmented data, and by compacting information. On a partitioned table, you can reorganize a single partition.

**Scope**

This command affects all database partitions in the database partition group.

**Authorization**

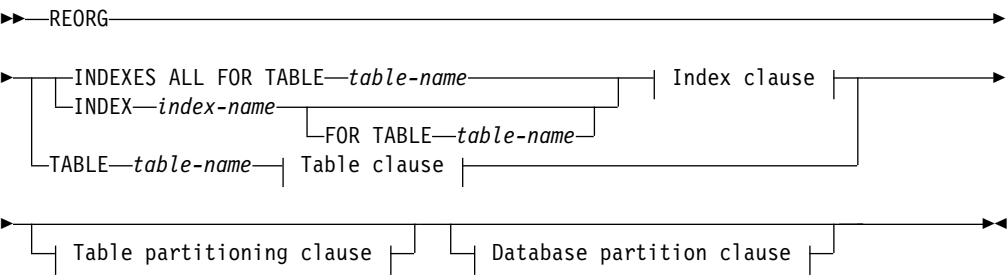
One of the following authorities:

- SYSADM
- SYSCtrl
- SYSMAINT
- DBADM
- SQLADM
- CONTROL privilege on the table.

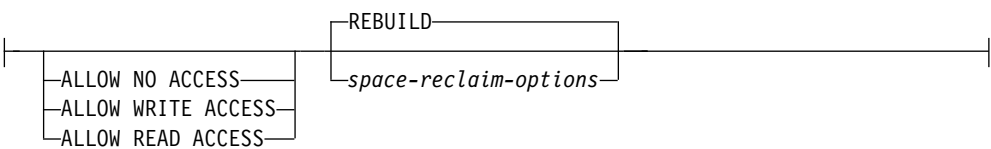
**Required connection**

Database

**Command syntax**



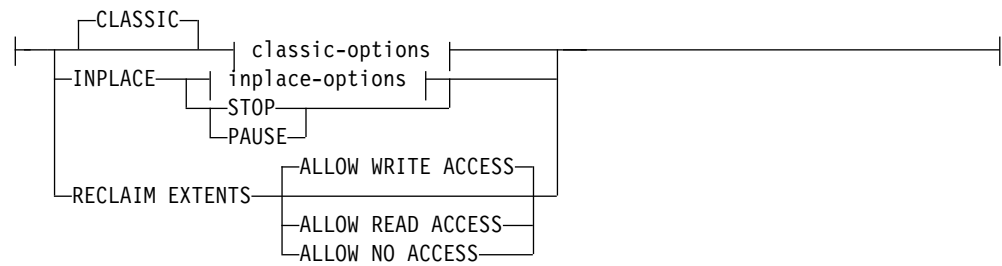
**Index clause:**



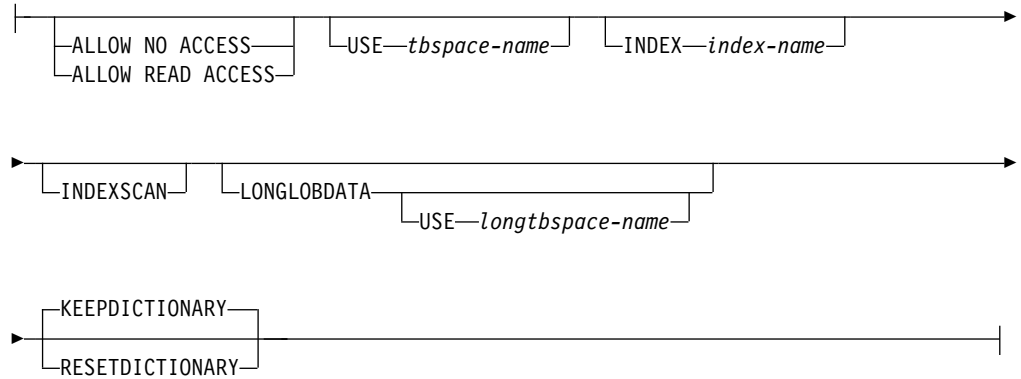
**space-reclaim-options:**



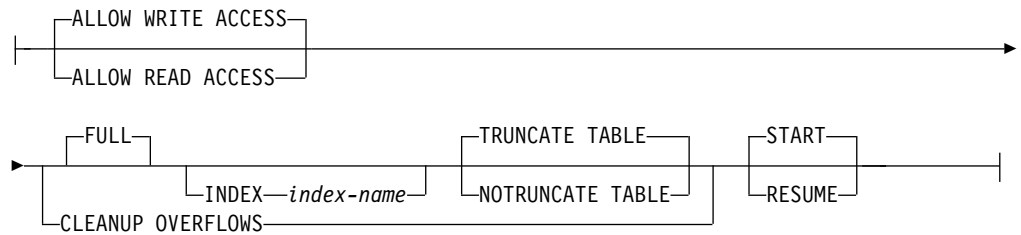
**Table clause:**



### classic-options:



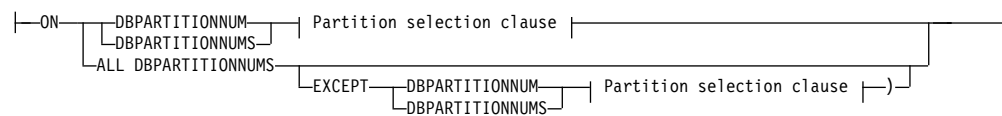
### inplace-options:



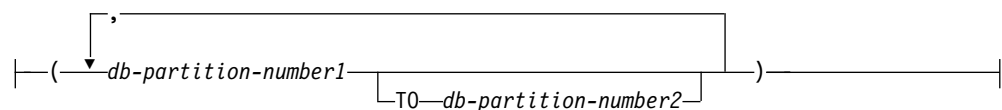
### Table partitioning clause:



### Database partition clause:



### Partition selection clause:



## Command parameters

### **INDEXES ALL FOR TABLE** *table-name*

Specifies the table whose indexes are to be reorganized. The table can be in a local or a remote database.

### **INDEX** *index-name*

Specifies an individual index to be reorganized.

Reorganization of individual indexes is supported in the following scenarios:

- Nonpartitioned indexes on a data partitioned table that are not block indexes
- Any index on any permanent table if **CLEANUP ALL** is specified and **RECLAIM EXTENTS** is not specified

### **FOR TABLE** *table-name*

Specifies the name of the table on which the index *index-name* is created. This parameter is optional, given that index names are unique across the database.

### **TABLE** *table-name*

Specifies the table to reorganize. The table can be in a local or a remote database. The name or alias in the form: *schema.table-name* can be used. The *schema* is the user name under which the table was created. If you omit the schema name, the default schema is assumed.

The **RECLAIM EXTENTS** parameter is the only parameter that is supported for column-organized tables.

For typed tables, the specified table name must be the name of the hierarchy's root table.

You cannot specify an index for the reorganization of a multidimensional clustering (MDC) or insert time clustering (ITC) table. Reorganization of tables cannot be used in place for MDC or ITC tables.

When the **ON DATA PARTITION** clause is specified for a table reorganization of a data partitioned table, only the specified data partition is reorganized:

- If there are no nonpartitioned indexes (except system-generated XML path indexes) defined on the table, the access mode applies only to the specified partition. Users are allowed to read from and write to the other partitions of the table.
- If there are nonpartitioned indexes that are defined on the table (excluding system-generated XML path indexes), the **ALLOW NO ACCESS** mode is the default and only supported access mode. In this case, the table is placed in **ALLOW NO ACCESS** mode. If **ALLOW READ ACCESS** is specified, SQL1548N is returned (SQLSTATE 5U047).

Table 22. Supported access mode for CLASSIC table reorganization on nonpartitioned and partitioned table

Command	Table type	Table partitioning clause	Supported access mode
<b>REORG TABLE</b>	Nonpartitioned table	Not applicable	<b>ALLOW NO ACCESS,</b> <b>ALLOW READ ACCESS</b> <sup>1</sup>
<b>REORG TABLE</b>	Partitioned table	Not specified	<b>ALLOW NO ACCESS</b> <sup>1</sup>
<b>REORG TABLE</b> (There are no indexes or only partitioned indexes that are defined on the table)	Partitioned table	<b>ON DATA PARTITION</b>	<b>ALLOW NO ACCESS,</b> <b>ALLOW READ ACCESS</b> <sup>1</sup>

Table 22. Supported access mode for CLASSIC table reorganization on nonpartitioned and partitioned table (continued)

Command	Table type	Table partitioning clause	Supported access mode
<b>REORG TABLE</b> (there are nonpartitioned indexes that are defined on the table, excluding system-generated XML path indexes)	Partitioned table	<b>ON DATA PARTITION</b>	<b>ALLOW NO ACCESS</b> <sup>1</sup>

**Note:**

1. Default mode when an access clause is not specified.

For a data partitioned table, a table reorganization rebuilds the nonpartitioned indexes and partitioned indexes on the table after the table is reorganized. If the **ON DATA PARTITION** clause is used to reorganize a specific data partition of a data partitioned table, a table reorganization rebuilds the nonpartitioned indexes and partitioned indexes only for the specified partition.

**Index clause**

**ALLOW NO ACCESS**

For **REORG INDEXES**, specifies that no other users can access the table while the indexes are being reorganized. If the **ON DATA PARTITION** clause is specified for a partitioned table, only the specified partition is restricted to the access mode level.

For **REORG INDEX**, specifies that no other users can access the table while the index is being reorganized.

**ALLOW READ ACCESS**

For **REORG INDEXES**, specifies that other users can have read-only access to the table while the indexes are being reorganized. **ALLOW READ ACCESS** mode is not supported for **REORG INDEXES** of a partitioned table unless the **CLEANUP** or **RECLAIM EXTENTS** option or the **ON DATA PARTITION** clause is specified. If the **ON DATA PARTITION** clause is specified for a partitioned table, only the specified partition is restricted to the access mode level.

For **REORG INDEX**, specifies that can have read-only access to the table while the index is being reorganized.

**ALLOW WRITE ACCESS**

For **REORG INDEXES**, specifies that other users can read from and write to the table while the indexes are being reorganized. **ALLOW WRITE ACCESS** mode is not supported for a partitioned table unless the **CLEANUP** or **RECLAIM EXTENTS** option or the **ON DATA PARTITION** clause is specified. If the **ON DATA PARTITION** clause is specified for a partitioned table, only the specified partition is restricted to the access mode level.

For **REORG INDEX**, specifies that can read from and write to the table while the index is being reorganized.

**ALLOW WRITE ACCESS** mode is not supported for multidimensional clustering (MDC) or insert time clustering (ITC) tables or extended indexes unless the **CLEANUP** or **RECLAIM EXTENTS** option is specified.

The **ALLOW WRITE ACCESS** parameter is not supported for column-organized tables if you specify the **REBUILD** parameter.

The following items apply for a data partitioned table when the **ON DATA PARTITION** clause is specified with the **REORG INDEXES ALL** command:

- Only the specified data partition is restricted to the access mode level. Users are allowed to read from and write to the other partitions of the table while the partitioned indexes of a specified partition are being reorganized.

The following table lists the access modes that are supported and the concurrent access that is allowed on other partitions of the table when the **ON DATA PARTITION** clause is specified:

*Table 23. Access modes supported and concurrent access allowed when the ON DATA PARTITION clause is specified with REORG INDEXES ALL*

Access mode	Concurrent access that is allowed on the specified partition	Concurrent access that is allowed on other partitions
<b>ALLOW NO ACCESS</b>	No access	Read and write access
<b>ALLOW READ ACCESS</b>	Read on the partition up until index is updated	Read and write access
<b>ALLOW WRITE ACCESS</b>	Read and write access on the partition up until index is updated	Read and write access

- Only the partitioned indexes for the specified partition are reorganized. The nonpartitioned indexes on the partitioned table are not reorganized.  
If there are any nonpartitioned indexes on the table marked "invalid" or "for rebuild", these indexes are rebuilt before reorganization. If not, only the partitioned indexes on the specified partition are reorganized or rebuilt if the index object is marked "invalid" or "for rebuild".
- Only partitioned indexes for the specified partition are cleaned when the **CLEANUP** or **RECLAIM EXTENTS** option is also specified.

The following table lists the supported access modes for index reorganization of partitioned and nonpartitioned tables:

*Table 24. Supported access modes for index reorganization on partitioned and nonpartitioned table*

Command	Table type	Table partitioning clause	Additional parameters that are specified for index clause	Supported access mode
<b>REORG INDEXES</b>	Nonpartitioned table	Not applicable	Any	<b>ALLOW NO ACCESS</b> , <b>ALLOW READ ACCESS</b> <sup>1</sup> , <b>ALLOW WRITE ACCESS</b> <sup>2</sup>
<b>REORG INDEX</b>	Nonpartitioned table	Not applicable	<b>CLEANUP ALL</b>	<b>ALLOW NO ACCESS</b> , <b>ALLOW READ ACCESS</b> <sup>1</sup> , <b>ALLOW WRITE ACCESS</b>
<b>REORG INDEX</b>	Partitioned table	Not applicable	Any	<b>ALLOW NO ACCESS</b> , <b>ALLOW READ ACCESS</b> <sup>1</sup> , <b>ALLOW WRITE ACCESS</b>
<b>REORG INDEX</b>	Partitioned table	With or without the <b>ON DATA PARTITION</b> clause	<b>CLEANUP ALL</b>	<b>ALLOW NO ACCESS</b> , <b>ALLOW READ ACCESS</b> <sup>1</sup> , <b>ALLOW WRITE ACCESS</b>
<b>REORG INDEXES</b>	Partitioned table	None	<b>REBUILD</b> (this is the default if none specified)	<b>ALLOW NO ACCESS</b> <sup>1</sup>

Table 24. Supported access modes for index reorganization on partitioned and nonpartitioned table (continued)

Command	Table type	Table partitioning clause	Additional parameters that are specified for index clause	Supported access mode
<b>REORG INDEXES</b>	Partitioned table	<b>ON DATA PARTITION</b>	<b>REBUILD</b> (this is the default if none specified)	<b>ALLOW NO ACCESS, ALLOW READ ACCESS<sup>1</sup>, ALLOW WRITE ACCESS</b>
<b>REORG INDEXES</b>	Partitioned table	With or without the <b>ON DATA PARTITION</b> clause	<b>CLEANUP</b> or <b>RECLAIM EXTENTS</b> specified	<b>ALLOW NO ACCESS, ALLOW READ ACCESS<sup>1</sup>, ALLOW WRITE ACCESS</b>

**Note:**

1. Default mode when an access clause is not specified.
2. **ALLOW WRITE ACCESS** is not supported for column-organized tables if you specify the **REBUILD** parameter.

Use the **ALLOW READ ACCESS** or **ALLOW WRITE ACCESS** option to allow other transactions either read-only or read-write access to the table while the indexes are being reorganized. No access to the table is allowed when you rebuild an index during the period in which the reorganized copies of the indexes are made available.

**REBUILD**

The **REBUILD** option is the default and represents the same functionality that is provided by index reorganization in previous releases when the **CLEANUP** and **CONVERT** clauses were not specified. The **REBUILD** option of index reorganization rebuilds the index data into physically contiguous pages. The default access mode depends on the table type.

**Space-reclaim-options**

**CLEANUP**

When **CLEANUP** is requested, a cleanup rather than a **REBUILD** is done. The indexes are not rebuilt and any pages that are freed up are available for reuse by indexes that are defined on this table only.

**ALL**

Specifies that indexes must be cleaned up by removing committed pseudo deleted keys and committed pseudo empty pages.

The **CLEANUP ALL** option frees committed pseudo empty pages, as well as remove committed pseudo deleted keys from pages that are not pseudo empty. This option also attempts to merge adjacent leaf pages if it results in a merged leaf page that has at least **PCTFREE** free space on the merged leaf page. **PCTFREE** is the percent free space that is defined for the index at index creation time. The default **PCTFREE** is 10 percent. If two pages can be merged, one of the pages are freed. The number of pseudo deleted keys in an index, except for the keys on pseudo empty pages, can be determined by running **RUNSTATS** and then selecting the **NUMRIDS DELETED** from **SYSCAT.INDEXES**. The **ALL** option cleans the **NUMRIDS DELETED** and the **NUM EMPTY LEAFS** if they are determined to be committed.

**PAGES**

Specifies that committed pseudo empty pages must be removed from the index tree. This step does not clean up pseudo deleted keys on pages that are not pseudo empty. Since it is checking only the pseudo empty leaf pages, it is considerably faster than using the **ALL** option in most cases.

The **CLEANUP PAGES** option searches for and free committed pseudo empty pages. A committed pseudo empty page is one where all the keys on the page are marked as deleted and all these deletions are known to be committed. The number of pseudo empty pages in an index can be determined by running **RUNSTATS** and looking at the NUM EMPTY LEAFs column in SYSCAT.INDEXES. The **PAGES** option cleans up the NUM EMPTY LEAFs if they are determined to be committed.

#### RECLAIM EXTENTS

Specifies the index to reorganize and reclaim extents that are not being used. This action moves index pages around within the index object to create empty extents. Then, this step frees these empty extents from exclusive use by the index object and makes the space available for use by other database objects within the table space. Extents are reclaimed from the index object back to the table space. **ALLOW READ ACCESS** is the default, but all access modes are supported.

#### Table clause

##### CLASSIC

Reorganizes the table by creating a new copy of the table and then replaces the original table with the reorganized copy. Any indexes on the table are recreated after the replacement. The original table might be available for queries until the replace operation starts, depending on the access clause, described next.

##### INPLACE

Reorganizes the table while it permits user access.

INPLACE table reorganization is allowed only on nonpartitioned, non-MDC, and non-ITC tables without extended indexes and with no indexes defined over XML columns in the table. Additionally, for a partitioned table, INPLACE table reorganization is allowed only if there are no nonpartitioned indexes (except system-generated XML path indexes) defined on the table and ON DATA PARTITION is specified. Only one data partition can be reorganized at a time. INPLACE table reorganization can be run only on tables that are at least three pages in size.

Before START can be issued for an INPLACE reorganization on a table, any paused or running in place reorganization must be completed or stopped on that table. For a partitioned table, if there is a paused or running in place reorganization on some partition, then before START can be issued for an INPLACE reorganization on another partition, the paused reorganization must be completed or stopped on that table.

In place table reorganization takes place asynchronously, and might not be effective immediately.

**STOP** Stop the in place **REORG** processing at its current point.

**PAUSE** Suspend or pause in place **REORG** at its current point.

#### RECLAIM EXTENTS

Specifies the table to reorganize and reclaim extents that are not being used. The *table-name* variable must specify a multidimensional clustering (MDC), insert time clustering (ITC) table, or column-organized tables. The name or alias in the form: *schema.table-name* can be used. The *schema* is the user name under which the table was created. If you omit the schema name, the default schema is assumed.

For **REORG TABLE RECLAIM EXTENTS** when the **ON DATA PARTITION** clause is specified, the access clause applies only to the named partition. Users can read from and write to the rest of the table while the extents on the specified partition are being reclaimed. This situation also applies to the default access levels.

#### **ALLOW NO ACCESS**

For **REORG TABLE RECLAIM EXTENTS**, specifies that no other users can access the table while the extents are being reclaimed.

#### **ALLOW READ ACCESS**

For **REORG TABLE RECLAIM EXTENTS**, specifies that other users can have read-only access to the table while the extents are being reclaimed.

#### **ALLOW WRITE ACCESS**

For **REORG TABLE RECLAIM EXTENTS**, specifies that other users can read from and write to the table while the extents are being reclaimed. This value is the default option.

### **classic options**

#### **ALLOW NO ACCESS**

Specifies that no other users can access the table while the table is being reorganized.

The **ALLOW NO ACCESS** mode is the default and only supported access mode when you reorganize a partitioned table without the **ON DATA PARTITION** clause.

If the **ON DATA PARTITION** clause is specified for a data partitioned table, only the specified data partition is reorganized:

- If there are no nonpartitioned indexes that are defined on the table (except system-generated XML path indexes), only the specified partition is restricted to the **ALLOW NO ACCESS** mode. Users are allowed to read from and write to the other partitions of the table.
- If there are nonpartitioned indexes that are defined on the table (except system-generated XML path indexes), the **ALLOW NO ACCESS** mode is the default and only supported access mode. In this case, the table is placed in **ALLOW NO ACCESS** mode.

#### **ALLOW READ ACCESS**

Allow only read access to the table during reorganization.

The **ALLOW READ ACCESS** mode is the default mode for a nonpartitioned table.

If the **ON DATA PARTITION** clause is specified for a data partitioned table, only the specified data partition is reorganized:

- If there are no nonpartitioned indexes that are defined on the table (except system-generated XML path indexes), the **ALLOW READ ACCESS** mode is the default mode. Only the specified partition is restricted to the access mode level. Users are allowed to read from and write to the other partitions of the table.
- If there are nonpartitioned indexes that are defined on the table (except system-generated XML path indexes), the **ALLOW READ ACCESS** mode is not supported. If **ALLOW READ ACCESS** is specified in this case, SQL1548N is returned (SQLSTATE 5U047).

**USE *tblspace-name***

Specifies the name of a system temporary table space in which to store a temporary copy of the table that is reorganized. If you do not provide a table space name, the database manager stores a working copy of the table in the table spaces that contain the table that is reorganized.

For an 8 KB, 16 KB, or 32 KB table object, if the page size of the system temporary table space that you specify does not match the page size of the table spaces in which the table data resides, the Db2 database product will try to find a temporary table space of the correct size of the LONG/LOB objects. Such a table space must exist for the reorganization to succeed.

For partitioned tables, the temporary table space is used as temporary storage for the reorganization of data partitions in the table. Reorganization of the entire partitioned table reorganizes a single data partition at a time. The temporary table space must be able to hold the largest data partition in the table, and not the entire table. When the **ON DATA PARTITION** clause is specified, the temporary table space must be able to hold the specified partition.

If you do not supply a table space name for a partitioned table, the table space where each data partition is located is used for temporary storage of that data partition. There must be enough free space in each data partition's table space to hold a copy of the data partition.

**INDEX *index-name***

Specifies the index to use when you reorganize the table. If you do not specify the fully qualified name in the form: *schema.index-name*, the default schema is assumed. The *schema* is the user name under which the index was created. The database manager uses the index to physically reorder the records in the table it is reorganizing.

For an in place table reorganization, if a clustering index is defined on the table and an index is specified, it must be the clustering index. If the in place option is not specified, any index that is specified is used. If you do not specify the name of an index, the records are reorganized without regard to order. However, if the table has a clustering index defined and no index is specified, then the clustering index is used to cluster the table. You cannot specify an index if you are reorganizing an MDC or ITC table.

If a table reorganization uses both the **INDEX** and **ON DATA PARTITION** clauses, only the specified partition is reorganized by using the index *index-name*.

**INDEXSCAN**

For a clustering **REORG**, an index scan is used to reorder table records. Reorganize table rows by accessing the table through an index. The default method is to scan the table and sort the result to reorganize the table by using temporary table spaces as necessary. Even though the index keys are in sort order, scanning and sorting is typically faster than fetching rows by first reading the row identifier from an index.

For tables with extended row size, the default method of scanning and sorting the result to reorganize the table is not supported.

For indexes that are expression-based, the default method of scanning and sorting the result to reorganize the table is not supported.

**LONGLOBDATA**

Long field and LOB data are to be reorganized.

This step is not required even if the table contains long or LOB columns. The default is to avoid reorganizing these objects because it is time consuming and does not improve clustering. However, running a reorganization with the **LONGLOBDATA** option on tables with XML columns reclaim unused space and reduce the size of the XML storage object.

This parameter is required when you convert existing LOB data into inlined LOB data.

For tables with extended row size, the first offline REORG after a table is altered enforces LONGLOBDATA.

#### **USE longtblspace-name**

This parameter is optional and can be used to specify the name of a temporary table space to be used for rebuilding long data. If no temporary table space is specified for either the table object or for the long objects, the objects are constructed in the table space they currently reside. If a temporary table space is specified for the table but this parameter is not specified, then the table space that is used for base reorg data is used, However, if the page sizes differ, the Db2 database system attempts to choose a temporary container of the appropriate page size to create the long objects in.

If **USE longtblspace-name** is specified, **USE tblspace-name** must also be specified. If it is not, the *longtblspace-name* argument is ignored.

#### **KEEPDICTIONARY**

If the COMPRESS attribute for the table is YES and the table has a compression dictionary then no new dictionary is built. All the rows that are processed during reorganization are subject to compression by using the existing dictionary. If the COMPRESS attribute is YES and a compression dictionary does not exist for the table, a dictionary is created (and the table compressed) in this scenario only if the table is of a certain size (approximately 1 - 2 MB) and sufficient data exists within this table. If, instead, you explicitly state **REORG RESETDICTIONARY**, then a dictionary is built as long as there is at least one row in the table. If the COMPRESS attribute for the table is NO and the table has a compression dictionary, then reorg processing preserves the dictionary. All the rows in the newly reorganized table are in non-compressed format. It is not possible to compress some data such as LOB data not stored in the base table row.

When the **LONGLOBDATA** option is not specified, only the table row data is reorganized. The following table describes the behavior of **KEEPDICTIONARY** syntax in **REORG** command when the **LONGLOBDATA** option is not specified.

Table 25. REORG KEEPDICTIONARY

Compress	Dictionary Exists	Result; outcome
Y	Y	Preserve dictionary; rows compressed.
Y	N	Build dictionary; rows compressed
N	Y	Preserve dictionary; all rows uncompressed
N	N	No effect; all rows uncompressed

The following table describes the behavior of **KEEPDICTIONARY** syntax in **REORG** command when the **LONGLOBDATA** option is specified.

Table 26. REORG KEEPDICTIONARY when LONGLOBDATA option is specified.

Compress	Table row data dictionary exists	XML storage object dictionary exists <sup>1</sup>	Compression dictionary	Data compression
Y	Y	Y	Preserve dictionaries.	Existing data is compressed. New data will be compressed.
Y	Y	N	Preserve table row dictionary and create an XML storage object dictionary.	Existing data is compressed. New data will be compressed.
Y	N	Y	Create table row dictionary and preserve the XML dictionary.	Existing data is compressed. New data will be compressed.
Y	N	N	Create table row and XML dictionaries.	Existing data is compressed. New data will be compressed.
N	Y	Y	Preserve table row and XML dictionaries.	Table data is uncompressed. New data will be not be compressed.
N	Y	N	Preserve table row dictionary.	Table data is uncompressed. New data will be not be compressed.
N	N	Y	Preserve XML dictionary.	Table data is uncompressed. New data will be not be compressed.
N	N	N	No effect.	Table data is uncompressed. New data will be not be compressed.

**Note:**

1. A compression dictionary can be created for the XML storage object of a table only if the XML columns are added to the table in Db2 V9.7 or later, or if the table is migrated by using the ADMIN\_MOVE\_TABLE stored procedure.

For any reinitialization or truncation of a table (such as for a replace operation), if the compress attribute for the table is NO, the dictionary is discarded if one exists. Conversely, if a dictionary exists and the compress attribute for the table is YES then a truncation saves the dictionary and not discard it. The dictionary is logged in its entirety for recovery purposes and for future support with data capture changes (that is, replication).

**RESETDICTIONARY**

If the COMPRESS attribute for the table is YES then a new row compression dictionary is built. All the rows that are processed during reorganization are subject to compression by using this new dictionary. This dictionary replaces any previous dictionary. If the COMPRESS attribute for the table is NO and the table does have an existing compression dictionary then reorg

processing removes the dictionary and all rows in the newly reorganized table will be in noncompressed format. It is not possible to compress some data such as LOB data not stored in the base table row.

If the **LONGLOBDATA** option is not specified, only the table row data is reorganized. The following table describes the behavior of **RESETDICTIONARY** syntax in **REORG** command when the **LONGLOBDATA** option is not specified.

Table 27. REORG RESETDICTIONARY

Compress	Dictionary Exists	Result; outcome
Y	Y	Build new dictionary*; rows compressed. If DATA CAPTURE CHANGES option is specified on the CREATE TABLE or ALTER TABLE statements, the current dictionary is kept (referred to as the <i>historical compression dictionary</i> ).
Y	N	Build new dictionary; rows compressed
N	Y	Remove dictionary; all rows uncompressed. If the DATA CAPTURE NONE option is specified on the CREATE TABLE or ALTER TABLE statements, the <i>historical compression dictionary</i> is also removed for the specified table.
N	N	No effect; all rows uncompressed

\* If a dictionary exists and the compression attribute is enabled but there currently isn't any data in the table, the **RESETDICTIONARY** operation keeps the existing dictionary. Rows that are smaller in size than the internal minimum record length and rows that do not demonstrate a savings in record length when an attempt is made to compress them are considered "insufficient" in this case.

The following table describes the behavior of **RESETDICTIONARY** syntax in **REORG** command when the **LONGLOBDATA** option is specified.

Table 28. REORG RESETDICTIONARY when LONGLOBDATA option is specified.

Compress	Table row data dictionary exists	XML storage object dictionary exists <sup>1</sup>	Data dictionary	Data compression
Y	Y	Y	Build dictionaries <sup>2 3</sup> .	Existing data is compressed. New data will be compressed.
Y	Y	N	Build new table row dictionary and create a new XML dictionary <sup>3</sup> .	Existing data is compressed. New data will be compressed.
Y	N	Y	Create table row data dictionary and build a new XML dictionary.	Existing data is compressed. New data will be compressed.
Y	N	N	Create dictionaries.	Existing data is compressed. New data will be compressed.
N	Y	Y	Remove dictionaries. Existing and new data is not compressed.	Existing table data is uncompressed. New data will be not be compressed.

Table 28. REORG RESETDICTIONARY when LONGLOBDATA option is specified. (continued)

Compress	Table row data dictionary exists	XML storage object dictionary exists <sup>1</sup>	Data dictionary	Data compression
N	Y	N	Remove table row dictionary. All data is uncompressed.	Existing table data is uncompressed. New data will be not be compressed.
N	N	Y	Remove XML storage object dictionary.	Existing table data is uncompressed. New data will be not be compressed.
N	N	N	No effect.	Existing table data is uncompressed. New data will be not be compressed.

**Note:**

1. A compression dictionary can be created for the XML storage object of a table only if the XML columns are added to the table in Db2 V9.7 or later, or if the table is migrated by using an online table move.
2. If a dictionary exists and the compression attribute is enabled but there currently isn't any data in the table, the **RESETDICTIONARY** operation keeps the existing dictionary. Rows that are smaller in size than the internal minimum record length and rows that do not demonstrate a savings in record length when an attempt is made to compress them are considered insufficient in this case.
3. If DATA CAPTURE CHANGES option is specified on the CREATE TABLE or ALTER TABLE statements, the current data dictionary is kept (referred to as the *historical compression dictionary*).

**inplace-options**

**ALLOW READ ACCESS**

Allow only read access to the table during reorganization.

**ALLOW WRITE ACCESS**

Allow write access to the table during reorganization. This is the default behavior.

**FULL** The table is reorganized to fill pages while respecting PCTFREE for the table. Optionally, when the INDEX clause is specified, row data is moved within the table to recluster the data. Overflow records are also converted to normal records as part of this process. This behavior is the default behavior.

**INDEX *index-name***

Specifies the index to use when you reorganize the table. If you do not specify the fully qualified name in the form: *schema.index-name*, the default schema is assumed. The *schema* is the user name under which the index was created. The database manager uses the index to physically reorder the records in the table it is reorganizing.

For an in place table reorganization, if a clustering index is defined on the table and an index is specified, it must be the clustering index. If the in place option is not specified, any index that is specified is used. If you do not specify the name of an index, the records are reorganized without regard to order. However, if the table has a clustering index defined and no index is specified, then the clustering index is used to cluster the table. You cannot specify an index if you are reorganizing an MDC or ITC table.

If a table reorganization uses both the **INDEX** and **ON DATA PARTITION** clauses, only the specified partition is reorganized by using the index *index-name*.

#### **TRUNCATE TABLE**

Reclaim all extents that are empty at the end of the table and return them to the table space. Space reclamation is run at the end of the in place reorganization. During truncation, the table is S-locked, preventing updates to the table. Updates are not possible even if **ALLOW WRITE ACCESS** is specified. This behavior is the default behavior.

#### **NOTRUNCATE TABLE**

This operation allows for write access through the entire reorganization when **ALLOW WRITE ACCESS** is specified. There is no attempt to reclaim space that is used by the table. Do not truncate the table after in place reorganization. During truncation, the table is S-locked.

**START** Start the in place **REORG** processing. Because this option is the default, this keyword is optional.

**RESUME** Continue or resume a previously paused in place table reorganization. When an online reorganization is resumed and you want the same options as when the reorganization was paused, you must specify those options again when you resume.

#### **CLEANUP OVERFLOWS**

An **INPLACE CLEANUP OVERFLOWS** reorganization traverses the table and searches for pointer or overflow records. Any record found is converted to a normal record by the operation. This operation improves performance for tables that have a significant number of pointer or overflow records. The operation does not result in a reduction of the size of the table.

#### **Table partitioning clause**

##### **ON DATA PARTITION** *partition-name*

For data partitioned tables, specifies the data partition for the reorganization.

For Db2 V9.7 Fix Pack 1 and later releases, the clause can be used with the **REORG INDEXES ALL** command to reorganize the partitioned indexes on a specific partition and the **REORG TABLE** command to reorganize data of a specific partition.

When you use the clause with a **REORG TABLE** or **REORG INDEXES ALL** command on a partitioned table, the reorganization fails and returns SQL2222N with reason code 1 if the partition *partition-name* does not exist for the specified table. The reorganization fails and returns SQL2222N with reason code 3 if the partition *partition-name* is in the attached or detached state.

If the **REORG INDEX** command is run with the **ON DATA PARTITION** clause for a nonpartitioned index, the reorganization fails and returns SQL2222N

with reason code 2. If the **REORG INDEX** command is run for a nonpartitioned index without the **CLEANUP ALL** clause, or if the **RECLAIM** clause is also specified, then the reorganization fails and returns SQL0270N with reason code 89.

The **REORG TABLE** command fails and returns SQL1549N (SQLSTATE 5U047) if the partitioned table is in the reorg pending state and there are nonpartitioned indexes that are defined on the table.

### Database partition

#### **ON DBPARTITIONNUM | ON DBPARTITIONNUMS**

Perform operation on a set of database partitions.

#### **ALL DBPARTITIONNUMS**

Specifies that operation is to be done on all database partitions that are specified in the `db2nodes.cfg` file. This option is the default if a database partition clause is not specified.

**EXCEPT** Specifies that operation is to be done on all database partitions that are specified in the `db2nodes.cfg` file, except those partitions specified in the database partition list.

### Partition selection clause

#### *db-partition-number1*

Specifies a database partition number in the database partition list.

#### *db-partition-number2*

Specifies the second database partition number so that all database partitions from *db-partition-number1* up to and including *db-partition-number2* are included in the database partition list.

### Examples

To reorganize a table to reclaim space and use the temporary table space `mytemp1`, enter the following command:

```
db2 reorg table homer.employee use mytemp1
```

To reorganize tables in a partition group that consists of database partitions 1, 2, 3, and 4 of a four-partition system, you can enter either of the following commands:

```
db2 reorg table employee index empid on dbpartitionnum (1,3,4)
```

```
db2 reorg table homer.employee index homer.empid on all
dbpartitionnums except dbpartitionnum (2)
```

To clean up the pseudo deleted keys and pseudo empty pages in all the indexes on the `EMPLOYEE` table while you allow other transactions to read and update the table, enter:

```
db2 reorg indexes all for table homer.employee allow write
access cleanup
```

To clean up the pseudo empty pages in all the indexes on the `EMPLOYEE` table while you allow other transactions to read and update the table, enter:

```
db2 reorg indexes all for table homer.employee allow write
access cleanup pages
```

To reorganize the `EMPLOYEE` table by using the system temporary table space `TEMPSPACE1` as a work area, enter:

```
db2 reorg table homer.employee use temp space1
```

To start, pause, and resume an in place reorg of the EMPLOYEE table with the default schema HOMER, which is specified explicitly in previous examples, enter the following commands:

```
db2 reorg table employee index empid inplace start
db2 reorg table employee inplace pause
db2 reorg table homer.employee inplace allow read access
notruncate table resume
```

The command to resume the reorg contains extra keywords to specify read access only and to skip the truncation step, which share-locks the table.

To clean up the pseudo-deleted keys and pseudo-empty pages in index EMPID on data partition PART1 of the data partitioned EMPLOYEE table, while allowing other transactions to read and update the table, enter:

```
db2 reorg index EMPID on table HOMER.EMPLOYEE
cleanup only all on data partition PART1
```

### Usage notes

Restrictions:

- The **REORG** utility does not support the use of nicknames.
- The **REORG TABLE** command is not supported for declared temporary tables or created temporary tables.
- The **REORG TABLE** command cannot be used on views.
- Reorganization of a table is not compatible with range-clustered tables because the range area of the table always remains clustered.
- **REORG TABLE** cannot be used on a partitioned table in a DMS table space while an online backup of ANY table space in which the table resides, including LOBs and indexes, is being performed.
- **REORG TABLE** cannot use an index that is based on an index extension.
- If a table is in reorg pending state, an in-place reorg is not allowed on the table.
- Concurrent table reorganization that share temporary DMS table space is not supported.
- If a table has an index with an expression-based key that is defined on it, in-place table reorganization is not supported.
- Before you run a reorganization operation against a table to which event monitors write, you need to deactivate the event monitors on that table.
- For data partitioned tables:
  - The table must have an ACCESS\_MODE in SYSCAT.TABLES of Full Access.
  - Reorganization skips data partitions that are in a restricted state due to an attach or detach operation. If the **ON DATA PARTITION** clause is specified, that partition must be fully accessible.
  - If an error occurs during table reorganization, some indexes or index partitions might be left invalid. The nonpartitioned indexes of the table is marked invalid if the reorganization has reached or passed the replace phase for the first data partition. The index partitions for any data partition that has already reached or passed the replace phase is marked invalid. Indexes will be rebuilt on the next access to the table or data partition.
  - If an error occurs during index reorganization when the **ALLOW NO ACCESS** mode is used, some indexes on the table might be left invalid. For nonpartitioned RID indexes on the table, only the index that is being

reorganized at the time of the failure is left invalid. For MDC tables with nonpartitioned block indexes, one or more of the block indexes might be left invalid if an error occurs. For MDC or ITC tables with partitioned indexes, only the index object on the data partition that is reorganized is left invalid. Any indexes marked invalid will be rebuilt on the next access to the table or data partition.

- When a data partitioned table with only partitioned indexes that are defined on the table is in the reorg pending state, issuing a **REORG TABLE** command with the **ON DATA PARTITION** clause brings only the specified data partition out of the reorg pending state. To bring the remaining partitions of the table out of the reorg pending state, either issue **REORG TABLE** command on the entire table (without the **ON DATA PARTITION** clause). You can also issue a **REORG TABLE** command with the **ON DATA PARTITION** clause for each of the remaining partitions.

Information about the current progress of table reorganization is written to the history file for database activity. The history file contains a record for each reorganization event. To view this file, execute the **LIST HISTORY** command for the database that contains the table you are reorganizing.

You can also use table snapshots to monitor the progress of table reorganization. Table reorganization monitoring data is recorded regardless of the Database Monitor Table Switch setting.

If an error occurs, an SQLCA dump is written to the history file. For an in-place table reorganization, the status is recorded as PAUSED.

When you modify an index table too many times, the data in the indexes might become fragmented. If the table is clustered with respect to an index, the table and index can get out of cluster order. Both of these factors can adversely affect the performance of scans that use the index, and can impact the effectiveness of index page prefetching. **REORG INDEX** or **REORG INDEXES** with the **REBUILD** option can be used to reorganize one or all of the indexes on a table. Index reorganization rebuild removes any fragmentation and restore physical clustering to the leaf pages. Use the **REORGCHK** command to help determine whether an index needs reorganizing. Be sure to complete all database operations and release all locks before you invoke index reorganization. This step can be done by issuing a **COMMIT** after you close all cursors opened **WITH HOLD**, or by issuing a **ROLLBACK**.

A classic table reorganization (offline reorganization) rebuilds the indexes during the last phase of the reorganization. When more than one temporary table space exists, it is possible that a temporary table space in addition to the one specified on the **REORG TABLE** command can be utilized for additional sorts that can accompany the table reorg process. However, the in-place table reorganization (online reorganization) does not rebuild the indexes. It is recommended that you issue a **REORG INDEXES** command after the completion of an in-place table reorganization. An in-place table reorganization is asynchronous. Therefore, care must be taken to ensure that the in-place table reorganization is complete before you issue the **REORG INDEXES** command. Issuing the **REORG INDEXES** command before the in-place table reorganization is complete, might cause the reorganization to fail (SQLCODE -2219).

When the REORG rebuilds the indexes on an MDC table, the **Full\_Block** hint bits are not set. Because the **Full\_Block** hint is not set, you might experience degraded insert performance if you insert rows from existing dimension values after the

REORG completes and the DB2\_TRUST\_MDC\_BLOCK\_FULL\_HINT registry variable is turned on. The insert performance automatically improves for each dimension value after an insert of that dimension value completes. For more information, see, DB2\_TRUST\_MDC\_BLOCK\_FULL\_HINT performance variable.

Tables that are modified so many times that data is fragmented and access performance is noticeably slow are candidates for the **REORG TABLE** command. You must also invoke this utility after you alter the inline length of a structured type column in order to benefit from the altered inline length. Use the **REORGCHK** command to determine whether a table needs reorganizing. Be sure to complete all database operations and release all locks before you invoke **REORG TABLE**. This step can be done by issuing a COMMIT after you close all cursors opened WITH HOLD, or by issuing a ROLLBACK. After you reorganize a table, use **RUNSTATS** to update the table statistics, and **REBIND** to rebind the packages that use this table. The reorganize utility implicitly closes all the cursors.

With Db2 V9.7 Fix Pack 1 and later, **REORG TABLE** commands that use the CLASSIC option and **REORG INDEXES ALL** commands can be issued on a data partitioned table to concurrently reorganize different data partitions or partitioned indexes on a partition. When concurrently reorganizing data partitions or the partitioned indexes on a partition, users can access the unaffected partitions but cannot access the affected partitions. All the following criteria must be met to issue **REORG** commands that operate concurrently on the same table:

- Each **REORG** command must specify a different partition with the **ON DATA PARTITION** clause.
- Each **REORG** command must use the **ALLOW NO ACCESS** mode restrict access to the data partitions.
- The partitioned table must have only partitioned indexes if you issue **REORG TABLE** commands. No nonpartitioned indexes (except system-generated XML path indexes) can be defined on the table.

For a partitioned table T1 with no nonpartitioned indexes (except system-generated XML path indexes) and with partitions P1, P2, P3, and P4, the following REORG commands can run concurrently:

```
REORG INDEXES ALL FOR TABLE T1 ALLOW NO ACCESS ON DATA PARTITION P1
REORG TABLE T1 ALLOW NO ACCESS ON DATA PARTITION P2
REORG INDEXES ALL FOR TABLE T1 ALLOW NO ACCESS ON DATA PARTITION P3
```

Operations such as the following are not supported when you use concurrent **REORG** commands:

- Using a **REORG** command without the **ON DATA PARTITION** clause on the table.
- Using an ALTER TABLE statement on the table to add, attach, or detach a data partition.
- Loading data into the table.
- Performing an online backup that includes the table.

If the table contains mixed row format because the table value compression is activated or deactivated, an offline table reorganization can convert all the existing rows into the target row format.

If the table is distributed across several database partitions, and the table or index reorganization fails on any of the affected database partitions, only the failing database partitions have the table or index reorganization rolled back.

If the reorganization is not successful, temporary files must not be deleted. The database manager uses these files to recover the database.

If the name of an index is specified, the database manager reorganizes the data according to the order in the index. To maximize performance, specify an index that is often used in SQL queries. If the name of an index is *not* specified, and if a clustering index exists, the data is ordered according to the clustering index.

The PCTFREE value of a table determines the amount of free space that is designated per page. If the value is not set, the utility fills up as much space as possible on each page.

To complete a table space rollforward recovery after a table reorganization, both regular and large table spaces must be enabled for rollforward recovery.

If the table contains LOB columns that do not use the **COMPACT** option, the LOB DATA storage object can be significantly larger following table reorganization. This step can be a result of the order in which the rows were reorganized, and the types of table spaces used (SMS or DMS).

Indexes over XML data might be recreated by the **REORG INDEXES/TABLE** command. For more information, see Recreation of indexes over XML data.

An in-place **REORG** operation might not be able to fully reclaim space in a table because it cannot move internal records.

An index is used internally to track pages within column-organized tables, and a modification state index might be created during index scans on these tables. Part of space management for column-organized tables is managing the space that is used by indexes, including the page map and modification state indexes. Reorganizing the table generates pseudo-deleted keys in the page map indexes. Update operations result in pseudo-deleted keys in the modification state index. Update and delete operations result in pseudo-deleted keys in regular indexes. Consider running a **REORG INDEXES** command, specifying the **CLEANUP** and **RECLAIM EXTENTS** parameters, or a **REORG INDEXES** command with the **CLEANUP** parameter, to clean up pseudo-deleted keys. Automatic table maintenance, if enabled, manages the cleanup and space reclamation process for indexes.

#### **RESET ALERT CONFIGURATION command using the ADMIN\_CMD procedure:**

Resets the health indicator settings for specific objects to the current defaults for that object type or resets the current default health indicator settings for an object type to the install defaults.

**Important:** This command or API has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at [http://www.ibm.com/support/knowledgecenter/SSEPGG\\_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html](http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html).

#### **Authorization**

One of the following authorities:

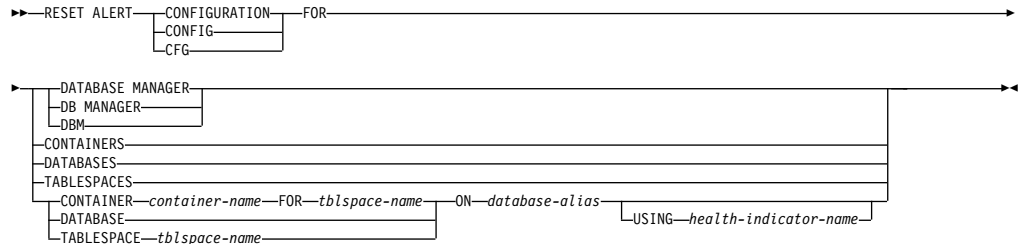
- SYSADM
- SYSMANT

- SYSCTRL

## Required connection

Instance. An explicit attachment is not required.

## Command syntax



## Command parameters

### DATABASE MANAGER | DB MANAGER | DBM

Resets alert settings for the database manager.

### CONTAINERS

Resets default alert settings for all table space containers managed by the database manager to the install default. These are the settings that apply to all table space containers that do not have custom settings. Custom settings are defined using the **CONTAINER** *container-name* **FOR** *tblspace-name* **ON** *database-alias* clause.

### DATABASES

Resets alert settings for all databases managed by the database manager. These are the settings that apply to all databases that do not have custom settings. Custom settings are defined using the **DATABASE** **ON** *database-alias* clause.

### TABLESPACES

Resets default alert settings for all table spaces managed by the database manager to the install default. These are the settings that apply to all table spaces that do not have custom settings. Custom settings are defined using the **TABLESPACE** *tblspace-name* **ON** *database-alias* clause.

### CONTAINER *container-name* **FOR** *tblspace-name* **ON** *database-alias*

Resets the alert settings for the table space container called *container-name*, for the table space specified using the **FOR** *tblspace-name* clause, on the database specified using the **ON** *database-alias* clause. If this table space container has custom settings, then these settings are removed and the current table space containers default is used.

### DATABASE **ON** *database-alias*

Resets the alert settings for the database specified using the **ON** *database-alias* clause. If this database has custom settings, then these settings are removed and the install default is used.

### TABLESPACE *tblspace-name* **ON** *database-alias*

Resets the alert settings for the table space called *tblspace-name*, on the database specified using the **ON** *database-alias* clause. If this table space has custom settings, then these settings are removed and the install default is used.

### USING *health-indicator-name*

Specifies the set of health indicators for which alert configuration will be reset. Health indicator names consist of a two-letter object identifier followed by a name that describes what the indicator measures. For example:

```
db.sort_privmem_util
```

If you do not specify this option, all health indicators for the specified object or object type will be reset.

### RESET DATABASE CONFIGURATION command using the ADMIN\_CMD procedure:

Resets the configuration of a specific database to the system defaults.

#### Scope

This command updates all members by default, except when **MEMBER** is specified to reset only one member.

#### Authorization

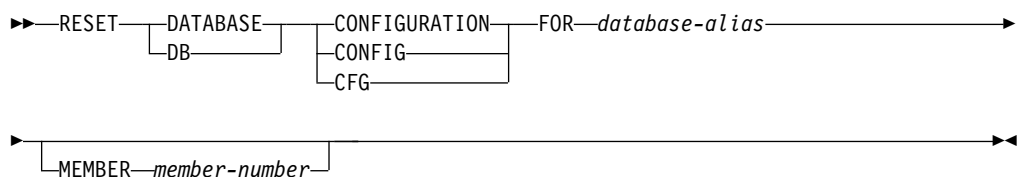
One of the following authorities:

- SYSADM
- SYSCTRL
- SYSMaint

#### Required connection

Instance. An explicit attachment is not required. If the database is listed as remote, an instance attachment to the remote node is established for the duration of the command.

#### Command syntax



#### Command parameters

##### FOR *database-alias*

Specifies the alias of the database whose configuration is to be reset to the system defaults.

##### MEMBER *member-number*

If a database configuration reset is to be applied to a specific member, this parameter may be used. If this parameter is not provided, the reset will take effect on all members.

## Example

### Reset the database configuration on a multi-partition instance

A user has a partitioned database environment that has 4 database partitions as defined in the `db2nodes.cfg`:

```
10 gilera 0
20 gilera 1
30 motobi 0
40 motobi 1
```

The user has created the SAMPLE database on the instance. The catalog partition for SAMPLE is on database partition number 10. Let's assume the user is logged on to system motobi.

The following command will reset all the database configuration values for database SAMPLE on all database partitions:

```
db2 reset db cfg for sample
```

The following commands will reset all the database configuration values for database SAMPLE, only at database partition number 30:

```
db2 reset db cfg for sample dbpartitionnum 30
```

or

```
export DB2NODE=30
db2 reset db cfg for sample
```

## Usage notes

To view or print a list of the database configuration parameters, use the **GET DATABASE CONFIGURATION** command.

To change the value of a configurable parameter, use the **UPDATE DATABASE CONFIGURATION** command.

Changes to the database configuration file become effective only after they are loaded into memory. All applications must disconnect from the database before this can occur.

If an error occurs, the database configuration file does not change.

The database configuration file cannot be reset if the checksum is invalid. This might occur if the database configuration file is changed without using the appropriate command. If this happens, the database must be restored to reset the database configuration file.

The **RESET DATABASE CONFIGURATION** command will reset the database configuration parameters to the documented default configuration values, where **auto\_runstats** will be ON. **Self\_tuning\_mem** will be reset to ON on non-partitioned database environments and to OFF on partitioned database environments.

The **RESET DATABASE CONFIGURATION** command does not change the database configuration parameters **ENCLIB** and **ENCROPTS**. The **RESET DATABASE CONFIGURATION** command is not done through a database connection, so Db2 is not able to determine whether a user holds SECADM authority.

## Compatibilities

For compatibility with previous versions:

- **DBPARTITIONNUM** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.

## RESET DATABASE MANAGER CONFIGURATION command using the ADMIN\_CMD procedure:

Resets the parameters in the database manager configuration file to the system defaults. The values are reset by node type.

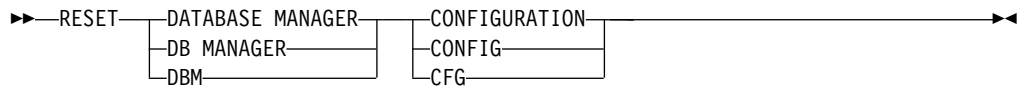
## Authorization

SYSADM

## Required connection

None or instance. An instance attachment is not required to perform local database manager configuration operations, but is required to perform remote database manager configuration operations. To update the database manager configuration for a remote instance, it is necessary to first attach to that instance. To update a configuration parameter online, it is also necessary to first attach to the instance.

## Command syntax



## Command parameters

None

## Usage notes

This command resets all parameters set by the installation program. This could cause error messages to be returned when restarting Db2. For example, if the **svccname** parameter is reset, the user will receive the SQL5043N error message when trying to restart Db2.

Before running this command, save the output from the **GET DATABASE MANAGER CONFIGURATION** command to a file so that you can refer to the existing settings. Individual settings can then be updated using the **UPDATE DATABASE MANAGER CONFIGURATION** command.

It is not recommended that the **svccname** parameter, set by the installation program, be modified by the user.

To view or print a list of the database manager configuration parameters, use the **GET DATABASE MANAGER CONFIGURATION** command. To change the value of a configurable parameter, use the **UPDATE DATABASE MANAGER CONFIGURATION** command.

For more information about these parameters, refer to the summary list of configuration parameters and the individual parameters.

Some changes to the database manager configuration file become effective only after they are loaded into memory. For more information about which parameters are configurable online and which ones are not, see the configuration parameter summary. Server configuration parameters that are not reset immediately are reset during execution of **db2start**. For a client configuration parameter, parameters are reset the next time you restart the application. If the client is the command line processor, it is necessary to invoke **TERMINATE**.

If an error occurs, the database manager configuration file does not change.

The database manager configuration file cannot be reset if the checksum is invalid. This might occur if you edit the configuration file manually and do not use the appropriate command. If the checksum is invalid, you must re-create the instance.

#### **REWIND TAPE command using the ADMIN\_CMD procedure:**

Rewinds tapes for backup and restore operations to streaming tape devices. This command is only supported on Windows operating systems.

#### **Authorization**

One of the following authorities:

- SYSADM
- SYSCtrl
- SYSMaint

#### **Required connection**

None

#### **Command syntax**

►►—REWIND TAPE—┐  
                    └ON—*device*—┘

#### **Command parameters**

##### **ON *device***

Specifies a valid tape device name. The default value is \\.\TAPE0.

#### **RUNSTATS command using the ADMIN\_CMD procedure:**

Updates statistics in the system catalog about the characteristics of a table, associated indexes, or statistical views. These characteristics include number of records, number of pages, and average record length. The optimizer uses these statistics to determine access paths to the data.

For a table, call the **RUNSTATS** command when the table had many updates, or after the table is reorganized. For a statistical view, call the **RUNSTATS** command when changes to underlying tables substantially affected the rows that are returned by the view. The view must be previously enabled for use in query optimization by using the ALTER VIEW statement.

## Scope

You can issue the **RUNSTATS** command from any database partition in the `db2nodes.cfg` file. You can use the command to update the catalogs on the catalog database partition.

For tables, this command collects statistics for a table on the database partition from which it is invoked. If the table does not exist on that database partition, the first database partition in the database partition group is selected.

For views, this command collects statistics by using data from tables on all participating database partitions.

## Authorization

For a table, you require one of the following authorities:

- SYSADM
- SYSCtrl
- SYSMAINT
- DBADM
- SQLADM
- CONTROL privilege on the table
- LOAD authority

You do not need any explicit privilege to use this command on any declared temporary table that exists within its connection.

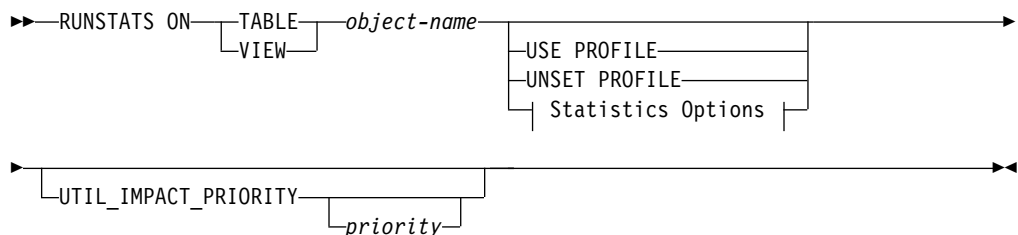
For statistical views, one of the following authorities:

- SYSADM
- SYSCtrl
- SYSMAINT
- DBADM
- SQLADM
- CONTROL privilege on the statistical view
- LOAD authority

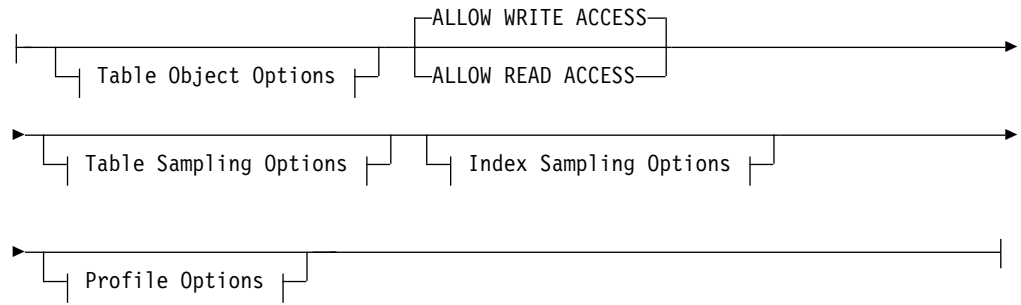
## Required connection

Database

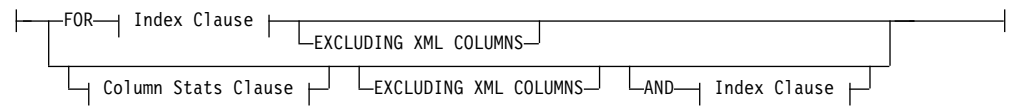
## Command syntax



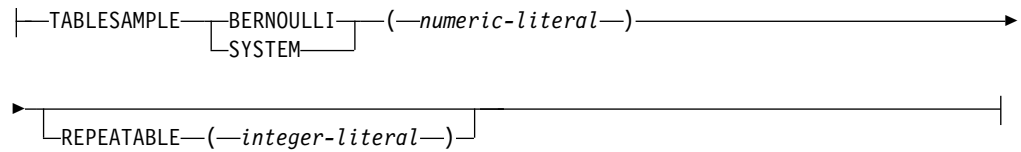
## Statistics Options:



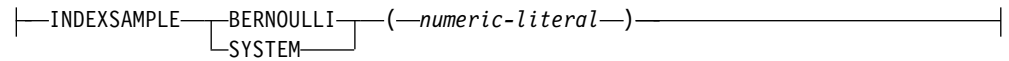
## Table Object Options:



## Table Sampling Options:



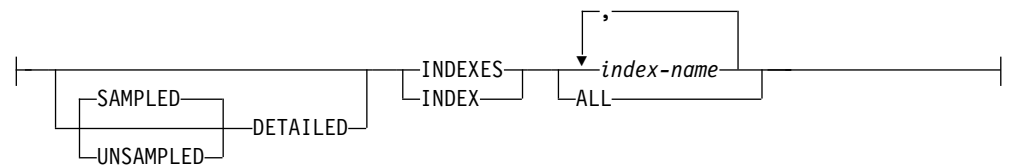
## Index Sampling Options:



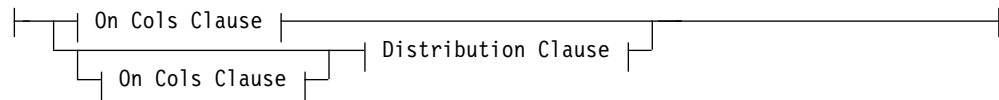
## Profile Options:



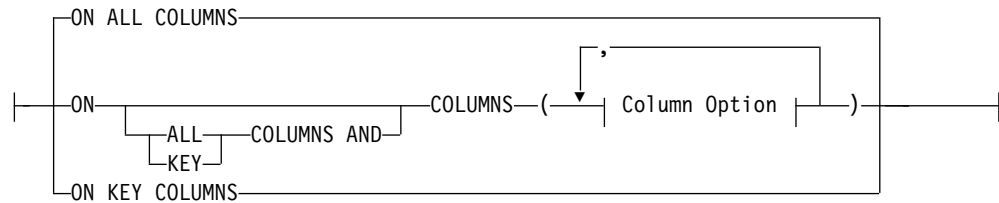
## Index Clause:



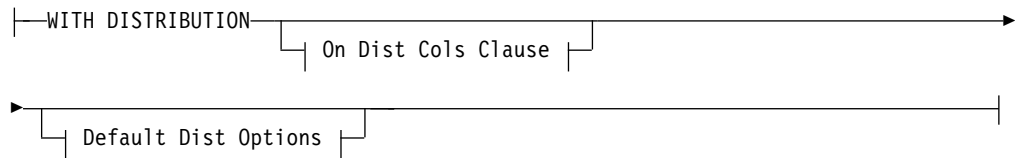
### Column Stats Clause:



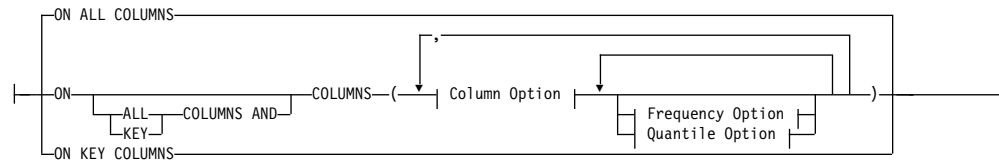
### On Cols Clause:



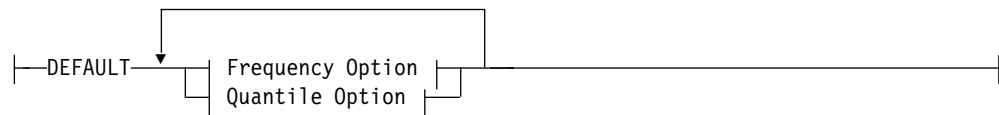
### Distribution Clause:



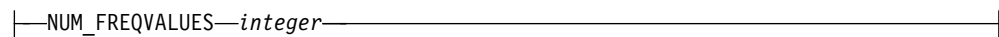
### On Dist Cols Clause:



### Default Dist Option:



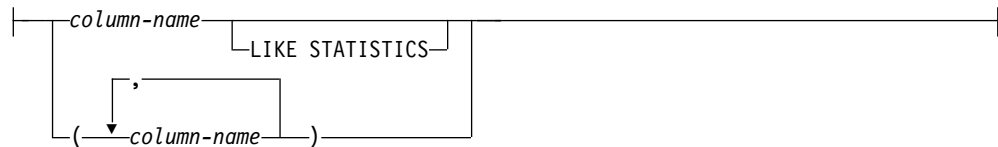
### Frequency Option:



### Quantile Option:



### Column Option:



## Command parameters

### *object-name*

Identifies the table or statistical view on which statistics are to be collected. This parameter must not be a table hierarchy. For typed tables, the value of the *object-name* parameter must be the name of the root table of the table hierarchy. The fully qualified name or alias in the form: *schema.object-name* must be used. The schema is the user name under which the table was created.

### USE PROFILE

This option allows **RUNSTATS** to employ a previously stored statistics profile to gather statistics for a table or statistical view. The statistics profile is created by using the **SET PROFILE** options and is updated by using the **UPDATE PROFILE** options.

### UNSET PROFILE

Removes a statistics profile. For example, the following command removes the profile for the `tablemyschema.mytable` table:

```
RUNSTATS ON tablemyschema.mytable UNSET PROFILE
```

### FOR INDEXES

Collects and updates statistics for the indexes only. If no table statistics was previously collected on the table, basic table statistics are also collected. Updates statistics for table cardinality (CARD), FPGES, NPAGES, even when table statistics exists. These basic statistics do not include any distribution statistics. This option cannot be used for views. COLCARD of the leading column of the index might also be updated.

### SAMPLED

Used together only with the **DETAILED** parameter. Specifying this option does not change the default functionality from **DETAILED**. The **SAMPLED** parameter is left in for compatibility with previous versions of Db2. This parameter cannot be used for views.

### UNSAMPLED

This option, when used with the **DETAILED** option, forces **RUNSTATS** to examine every entry in the index to compute the extended index statistics. You cannot use the **UNSAMPLED** parameter for views and it cannot be used together with scan index sampling (**INDEXSAMPLE** keyword). This option significantly increases **RUNSTATS** resource consumption, while rarely providing significant improvement over the **DETAILED** or **SAMPLED DETAILED** options, which are equivalent.

### DETAILED

Calculates extended index statistics. The extended index statistics are the CLUSTERFACTOR and PAGE\_FETCH\_PAIRS statistics that are gathered for relatively large indexes. Not all index entries are examined. A CPU sampling technique is employed instead to improve performance. You cannot use this parameter for views.

### *index-name*

Identifies an existing index that is defined on the table. If you do not specify the fully qualified name in the form: *schema.index-name*, the default schema is assumed.

### **EXCLUDING XML COLUMNS**

Omits all XML-type columns from statistics collection. Using this clause facilitates the collection of statistics on non-XML columns because the inclusion of XML data can require greater system resources. The **EXCLUDING XML COLUMNS** clause takes precedence over other clauses that specify XML columns for statistics collection. For example, if you use the **EXCLUDING XML COLUMNS** clause, and you also specify XML type columns with the **ON COLUMNS** clause or you use the **ON ALL COLUMNS** clause, all XML-type columns are ignored during statistics collection. For Db2 V9.7 Fix Pack 1 and later releases, distribution statistics over XML-type columns are not collected when you specify this parameter.

### **AND INDEXES**

Collects and updates statistics for both the table and the indexes.

### **ON ALL COLUMNS**

Collects statistics on all eligible columns. You can use this parameter with the **On Col s** clause or use this parameter with the **On Dist Col s** clause after the **WITH DISTRIBUTION** parameter. The **ON ALL COLUMNS** parameter is the default if you do not specify either of the column-specific clauses.

If it is specified in the **On Col s** clause, all columns have only basic column statistics collected unless specific columns are chosen as part of the **WITH DISTRIBUTION** clause. Those columns that are specified as part of the **WITH DISTRIBUTION** clause will also have basic and distribution statistics collected.

If the **WITH DISTRIBUTION ON ALL COLUMNS** is specified, both basic statistics and distribution statistics are collected for all eligible columns. Therefore, anything specified in the clause is redundant and not **On Col s** necessary.

### **ON COLUMNS**

To collect statistics on specific columns, column groups, or both, use the **ON COLUMNS**. A column group is a parenthesized comma-separated list of columns for which you want to collect combined statistics.

The column and column groups are specified as a parenthesized comma-separated list.

Running the **RUNSTATS** command on a table without gathering index statistics but specifying a subset of columns for which statistics are to be gathered has the following effects:

- Statistics are not reset for columns that you do not specify for the **RUNSTATS** command but are the first column in an index.
- Statistics are reset for all other columns that you do not specify for the **RUNSTATS** command.

You can use the **ON COLUMNS** parameter in the **On Col s** clause and the **On Dist Col s** clause. Collecting distribution statistics for a group of columns is not currently supported.

If you specify XML-type columns in a column group, the XML-type columns are ignored for collecting distinct values for the group. However, basic XML column statistics are collected for the XML-type columns in the column group.

#### **ON KEY COLUMNS**

Collects statistics on columns that make up all the indexes that are defined on the table. It is assumed that critical columns in queries are also those that are used to create indexes on the table. If no indexes on the table exist, no column statistics are collected.

You can use the **ON KEY COLUMNS** parameter in the `On Cols` clause or the `On Dist Cols` clause. Specifying the parameter in both clauses is redundant because if you specify the `On Dist Cols` clause (after the **WITH DISTRIBUTION** parameter), both basic and distribution statistics are collected.

XML-type columns are by definition not key columns and are not included for statistics collection by the **ON KEY COLUMNS** parameter. You cannot use this parameter for views.

#### *column-name*

Name of a column in the table or statistical view. If you specify the name of an ineligible column for statistics collection, such as a nonexistent column or a mistyped column name, error (-205) is returned. Two lists of columns can be specified, one without distribution and one with distribution. If the column is specified in the list that is not associated with the **WITH DISTRIBUTION** clause, only basic column statistics are collected. If the column appears in both lists, distribution statistics are collected (unless **NUM\_FREQVALUES** and **NUM\_QUANTILES** are set to zero).

#### **LIKE STATISTICS**

Collects additional column statistics for columns of type **BINARY**, **VARBINARY**, **CHAR**, or **VARCHAR** with a code page attribute of single-byte character set (SBCS), **FOR BIT DATA**, or **UTF-8**. The statistics are collected if the **runstats** utility determines that such statistics are appropriate after it analyzes column values. These statistics are shown in the **SUB\_COUNT** and the **SUB\_DELIM\_LENGTH** columns in the **SYSSTAT.COLUMNS** views. The query optimizer uses these statistics to improve the selectivity estimates for predicates of the type "column LIKE '%xyz'" and "column LIKE '%xyz%'".

#### **WITH DISTRIBUTION**

Collects both basic statistics and distribution statistics on columns. If you do not specify the **ON COLUMNS** parameter, distribution statistics are collected on all the columns of the table or statistical view, excluding columns that are ineligible such as columns of type **CLOB** and **LONG VARCHAR**. If you specify the **ON COLUMNS** parameter, distribution statistics are collected only on the column list that you provide, excluding those columns that are ineligible for statistics collection.

If you specify the **WITH DISTRIBUTION** parameter followed by the **ON COLUMNS** parameter with column groups, distribution statistics are not collected for the column groups.

#### **DEFAULT**

If you specify the **NUM\_FREQVALUES** and **NUM\_QUANTILES** parameters, the values of the parameters are used to determine the maximum number of frequency and quantile statistics to be collected for the columns. The **NUM\_FREQVALUES** and **NUM\_QUANTILES** parameters are used if you do not specify values for individual columns in the **ON COLUMNS** clause. If you do not specify the **DEFAULT** parameter, the values that are used are the values in the corresponding database configuration parameters.

#### NUM\_FREQVALUES

Defines the **DEFAULT** parameter referred to frequency statistics. You can specify this parameter for an individual column after the **ON COLUMNS** parameter. If you instead specify the **NUM\_FREQVALUES** parameter after the **DEFAULT** parameter, the value for the **NUM\_FREQVALUES** parameter after the **DEFAULT** parameter is used. If the specified value is **'-1'** or you do not specify the **NUM\_FREQVALUES** parameter after either the **ON COLUMNS** or **DEFAULT** parameter, the maximum number of frequency values is the value of the **num\_freqvalues** database configuration parameter.

#### NUM\_QUANTILES

Defines the maximum number of distribution quantile values to collect. It can be specified for an individual column in the **ON COLUMNS** clause. If the value is either not specified or is specified as **'-1'** for an individual column, the quantile limit value is picked up from the value specified in the **DEFAULT** clause. If it is not specified there either, the maximum number of quantile values to be collected are what is set in the **num\_quantiles** database configuration parameter.

For Db2 V9.7 Fix Pack 1 and later releases, distribution statistics for each index over XML data uses a maximum of 250 quantiles as the default. The default can be changed by specifying the **NUM\_QUANTILES** parameter in the **ON COLUMNS** or the **DEFAULT** clause. The **num\_quantiles** database configuration parameter is ignored while it collects XML distribution statistics.

#### ALLOW WRITE ACCESS

Specifies that other users can read from and write to the tables while statistics are calculated. For statistical views, these are the base tables that are referenced in the view definition.

The **ALLOW WRITE ACCESS** option is not recommended for tables that have numerous inserts, updates, or deletes occurring concurrently. The **RUNSTATS** command first collects table statistics and then performs index statistics. Changes in the table's state between the time that the table and index statistics are collected might result in inconsistencies. Although having up-to-date statistics is important for the optimization of queries, it is also important to have consistent statistics. Therefore, statistics must be collected at a time when inserts, updates, or deletes are at a minimum.

#### ALLOW READ ACCESS

Specifies that other users can have read-only access to the tables while statistics are calculated. For statistical views, these are the base tables that are referenced in the view definition.

#### TABLESAMPLE BERNOULLI

Collects statistics on a sample of the rows from the table or statistical view. *Bernoulli sampling* considers each row individually, including the row with probability  $P/100$  (where  $P$  is the value of the *numeric-literal* parameter) and excluding it with probability  $1-P/100$ . Thus, if the value of the *numeric-literal* parameter is evaluated to be the value 10, representing a 10% sample, each row would be included with probability 0.1 and be excluded with probability 0.9. Unless you specify the optional **REPEATABLE** parameter, each execution of the **RUNSTATS** command usually yields a different sample of the table. All data pages are retrieved through a table scan, but only the percentage of rows that you specify by using the *numeric-literal* parameter is used for statistics collection.

## **TABLESAMPLE SYSTEM**

Collects statistics on a sample of the data pages from the tables. *System sampling* considers each page individually, including the page with probability  $P/100$  (where  $P$  is the value of the *numeric-literal* parameter) and excluding it with probability  $1-P/100$ . Unless you specify the optional **REPEATABLE** parameter, each execution of the **RUNSTATS** command usually yields a different sample of the table. You control the size of the sample by specifying the *numeric-literal* parameter in parentheses, which represents an approximate percentage  $P$  of the table to return. Only a percentage of the data pages, as specified by the *numeric-literal* parameter, is retrieved and used for statistics collection.

For a statistical view, you can apply system sampling to only a single base table that is referenced in the view definition. If the view contains multiple tables, system sampling is possible if a single table in the statistical view can be identified as being joined with all primary keys or unique index columns of the other tables that are used in the view. If the statistical view does not meet those conditions, Bernoulli sampling is to be used instead, and a warning is returned.

### *numeric-literal*

Specifies the size of the sample to be obtained, as a percentage  $P$ . This value must be a positive number that is less than or equal to 100 and can be from 0 - 1. For example, a value of 0.01 represents one 1/100 of a percent, such that one row in 10,000 is sampled, on average. A value of 0 or 100 is treated as if you did not specify sampling, regardless of whether you specified the **TABLESAMPLE BERNOULLI** or **TABLESAMPLE SYSTEM** parameter. A value greater than 100 or less than 0 is treated as an error (SQL1197N).

### **REPEATABLE** (*integer-literal*)

When specified after the **TABLESAMPLE** parameter, ensures that repeated executions of the **RUNSTATS** command return the same sample. The *integer-literal* parameter specifies a non-negative integer that represents the seed to be used in sampling. Passing a negative seed results in an error (SQL1197N). The sample set might vary between repeatable **RUNSTATS** command invocations if activity against the table or statistical view that is resulted in changes to the table or statistical view data since the last time that you ran the command with the **TABLESAMPLE REPEATABLE** parameter. Also, to ensure consistent results, the method by which you obtained the sample (by using the **BERNOULLI** or **SYSTEM** parameter) must be the same.

## **INDEXSAMPLE BERNOULLI**

Collects index statistics on a sample of the rows in the index. *Bernoulli sampling* considers each row individually, including the row with probability  $P/100$  (where  $P$  is the value of the **numeric-literal** parameter) and excluding it with probability  $1-P/100$ . Thus, if the **numeric-literal** parameter has the value 10, representing 10%, each row would be included with probability 0.1 and be excluded with probability 0.9. Each execution of the **RUNSTATS** command is likely to yield a different sample of the index. All index pages are retrieved through an index scan, but only the percentage of rows as specified through the **numeric-literal** parameter is used for statistics collection. The **INDEXSAMPLE BERNOULLI** parameter is not supported for statistical views.

## **INDEXSAMPLE SYSTEM**

Collects statistics on a sample of the index pages. *System sampling* considers each page individually, including the page with probability  $P/100$  (where  $P$

is the value of the **numeric-literal** parameter) and excluding it with probability  $1-P/100$ . Each execution of the **RUNSTATS** command usually yields a different sample of the index. You control the size of the sample by specifying the **numeric-literal** parameter in parentheses, which represents an approximate percentage  $P$  of the index to return. Only a percentage of the index pages, as specified by the **numeric-literal** parameter, is retrieved and used for statistics collection. The **INDEXSAMPLE SYSTEM** parameter is not supported for statistical views.

#### **SET PROFILE NONE**

Specifies that no statistics profile is set for this **RUNSTATS** invocation.

#### **SET PROFILE**

Generates and stores a specific statistics profile in the system catalog tables and executes the **RUNSTATS** command options to gather statistics.

#### **SET PROFILE ONLY**

Generates and stores a specific statistics profile in the system catalog tables without running the **RUNSTATS** command options.

#### **UPDATE PROFILE**

Modifies a statistics profile in the system catalog tables and runs the **RUNSTATS** command options of the updated statistics profile to gather statistics. You cannot use the **UPDATE PROFILE** parameter to remove clauses that are in a statistics profile.

#### **UPDATE PROFILE ONLY**

Modifies a statistics profile in the system catalog tables without running the **RUNSTATS** command options of the updated statistics profile. You cannot use the **UPDATE PROFILE ONLY** parameter to remove clauses that are in a statistics profile.

#### **UTIL\_IMPACT\_PRIORITY** *priority*

Specifies that **RUNSTATS** is going to be throttled at the level that is specified by *priority*. *priority* is a number in the range of 1 to 100, with 100 representing the highest priority and 1 representing the lowest. The *priority* specifies the amount of throttling to which the utility is subjected. All utilities at the same *priority* undergo the same amount of throttling, and utilities at lower *priorities* are throttled more than utilities at higher *priorities*. If *priority* is not specified, the **RUNSTATS** has the default *priority* of 50. Omitting the **UTIL\_IMPACT\_PRIORITY** keyword invokes the **RUNSTATS** utility without throttling support. If the **UTIL\_IMPACT\_PRIORITY** keyword is specified, but the **util\_impact\_lim** configuration parameter is set to 100, then the utility runs unthrottled.

When you use the **RUNSTATS** command on tables in a partitioned database, statistics are collected on only a single database partition. If the database partition from which you ran the **RUNSTATS** command has a partition of the table, the command runs on that database partition. Otherwise, the command runs on the first database partition in the database partition group across which the table is partitioned.

#### **Examples**

- The following command collects basic statistics (no distribution statistics) on the table only, on all columns:  
**RUNSTATS ON TABLE emp**loyee
- The following command collects statistics on the table only, with distribution statistics on columns empid and empname:

```
RUNSTATS ON TABLE employee
 WITH DISTRIBUTION ON COLUMNS (empid, empname)
```

- The following command collects basic and distribution statistics on the table only, on all columns. The command specifies a maximum number of frequency values and uses the maximum number of distribution quantile values from the configuration setting.

```
RUNSTATS ON TABLE employee WITH DISTRIBUTION DEFAULT
 NUM_FREQVALUES 50
```

- The following command collects basic and distribution statistics on the table only, on all columns and on column group JOB, WORKDEPT, and SEX:

```
RUNSTATS ON TABLE employee ON ALL COLUMNS AND COLUMNS ((JOB, WORKDEPT, SEX))
 WITH DISTRIBUTION
```

- The following command collects statistics on a set of indexes:

```
RUNSTATS ON TABLE employee for indexes emp11, emp12
```

- The following command collects statistics on all indexes only:

```
RUNSTATS ON TABLE employee FOR INDEXES ALL
```

- The following command collects basic statistics on the table and all indexes, by using sampling for the detailed index statistics collection:

```
RUNSTATS ON TABLE employee AND SAMPLED DETAILED INDEXES ALL
```

This command is equivalent to the following one:

```
RUNSTATS ON TABLE employee AND DETAILED INDEXES ALL
```

- The following command collects basic statistics on the table, with distribution statistics on columns empid, empname, and empdept and the two indexes Xempid and Xempname. Distribution statistics limits are set individually for empdept, and the other two columns use a common default.

```
RUNSTATS ON TABLE employee
 WITH DISTRIBUTION ON COLUMNS (empid, empname, empdept NUM_FREQVALUES
 50 NUM_QUANTILES 100)
 DEFAULT NUM_FREQVALUES 5 NUM_QUANTILES 10
 AND INDEXES Xempid, Xempname
```

- The following command collects statistics on all columns that are used in indexes and on all indexes:

```
RUNSTATS ON TABLE employee ON KEY COLUMNS AND INDEXES ALL
```

- The following command collects statistics on all indexes and all columns with distribution and distribution statistics set individually for one of the columns. In this example, T1 contains columns c1, c2,..., c8

```
RUNSTATS ON TABLE T1
 WITH DISTRIBUTION ON COLUMNS (c1, c2, c3 NUM_FREQVALUES 20
 NUM_QUANTILES 40, c4, c5, c6, c7, c8)
 DEFAULT NUM_FREQVALUES 0, NUM_QUANTILES 0 AND INDEXES ALL
```

```
RUNSTATS ON TABLE T1
 WITH DISTRIBUTION ON COLUMNS (c3 NUM_FREQVALUES 20 NUM_QUANTILES 40)
 AND INDEXES ALL
```

- The following command collects statistics on table T1 for the individual columns c1 and c5 and for the column groups (c2, c3) and (c2, c4). Multicolumn cardinality is useful to the query optimizer when it estimates filter factors for predicates on columns in which the data is correlated.

```
RUNSTATS ON TABLE T1 ON COLUMNS (c1, (c2, c3),
 (c2, c4), c5)
```

- The following command collects statistics on table T1 for the individual columns c1 and c2. For column c1, the command also collects LIKE predicate statistics.

```
RUNSTATS ON TABLE T1 ON COLUMNS (c1 LIKE STATISTICS, c2)
```

- The following command registers a statistics profile to collect basic and distribution statistics on the table only, on all columns. The command specifies a maximum number of frequency values and uses the maximum number of distribution quantile values from the configuration setting.

```
RUNSTATS ON TABLE employee WITH DISTRIBUTION DEFAULT
NUM_FREQVALUES 50 SET PROFILE
```

- The following command registers a statistics profile to collect basic and distribution statistics on the table only, on all columns. The command specifies a maximum number of frequency values and uses the maximum number of distribution quantile values from the configuration setting. Statistics are not collected.

```
RUNSTATS ON TABLE employee WITH DISTRIBUTION
DEFAULT NUM_FREQVALUES 50 SET PROFILE ONLY
```

- The following command modifies the previously registered statistics profile by changing the maximum number of frequency values in the range 50 - 30. The command also updates the statistics as specified.

```
RUNSTATS ON TABLE employee WITH DISTRIBUTION
DEFAULT NUM_FREQVALUES 30 UPDATE PROFILE
```

- The following command modifies the previously registered statistics profile by changing the maximum number of frequency values in the range 50 - 30. Statistics are not collected.

```
RUNSTATS ON TABLE employee WITH DISTRIBUTION
DEFAULT NUM_FREQVALUES 30 UPDATE PROFILE ONLY
```

- The following command modifies the previously registered statistics profile by adding column empl\_address and column group (empl\_title, empl\_salary) options. The command also updates the statistics as specified.

```
RUNSTATS ON TABLE employee
ON COLUMNS (empl_address, (empl_title, empl_salary))
UPDATE PROFILE
```

- The following command modifies the previously registered statistics profile by adding column empl\_address and column group (empl\_title, empl\_salary) options. Statistics are not collected.

```
RUNSTATS ON TABLE employee
ON COLUMNS (empl_address, (empl_title, empl_salary))
UPDATE PROFILE ONLY
```

- The following command collects statistics on a table by using the options that are recorded in the statistics profile for that table:

```
RUNSTATS ON TABLE employee USE PROFILE
```

- The following command queries the **RUNSTATS** command options corresponding to the previously registered statistics profile in the catalogs of the table:

```
SELECT STATISTICS_PROFILE FROM SYSCAT.TABLES WHERE TABNAME =
'EMPLOYEE'
```

- The following command collects statistics, including distribution statistics, on 30% of the rows:

```
RUNSTATS ON TABLE employee WITH DISTRIBUTION
TABLESAMPLE BERNOULLI(30)
```

- To control the sample set on which statistics are collected and to be able to repeatedly use the same sample set, you can do so as follows:

```
RUNSTATS ON TABLE employee WITH DISTRIBUTION
TABLESAMPLE BERNOULLI(30) REPEATABLE(4196)
```

The command results in the same set of statistics if the data is not changed in the interim.

- The following command collects index statistics and table statistics on 1.5% of the data pages. Only table data pages, not index pages, are sampled. In this example, 1.5% of table data pages are used for the collection of table statistics, and all index pages are used for index statistics.  

```
RUNSTATS ON TABLE employee AND INDEXES ALL TABLESAMPLE SYSTEM(1.5)
```
- The following command collects table statistics on 1.5% of the data pages and index statistics on 2.5% of the index pages. Both table data pages and index pages are sampled.  

```
RUNSTATS ON TABLE employee AND INDEXES ALL TABLESAMPLE SYSTEM(1.5)
INDEXSAMPLE SYSTEM(2.5)
```
- The following command collects statistics for a statistical view, on all columns, without distribution statistics:  

```
RUNSTATS ON VIEW product_sales_view
```
- The following command collects basic statistics for a statistical view, with distribution statistics on the category, type and product\_key columns. Distribution statistics limits are set for the category column, and the other columns use a common default.  

```
RUNSTATS ON VIEW product_sales_view
WITH DISTRIBUTION ON COLUMNS (category NUM_FREQVALUES 100 NUM_QUANTILES 100,
type, product_key) DEFAULT NUM_FREQVALUES 50 NUM_QUANTILES 50
```
- The following command collects basic and distribution statistics for a statistical view on 10% of the rows, by using row-level sampling:  

```
RUNSTATS ON VIEW product_sales_view
WITH DISTRIBUTION TABLESAMPLE BERNOULLI (10)
```
- The following command collects basic and distribution statistics for a statistical view on 2.5% of the rows, by using data-page-level sampling. Additionally, this command specifies the repeated use of the same sample set. For this command to succeed, the query must be such that the Db2 database system can successfully push data-page sampling down to one or more tables. Otherwise, an error (SQL 20288N) is issued.  

```
RUNSTATS ON VIEW product_sales_view
WITH DISTRIBUTION TABLESAMPLE SYSTEM (2.5) REPEATABLE(4196)
```
- The following command registers a statistics profile to collect statistics on the view and on all columns with distribution statistics as specified:  

```
RUNSTATS ON VIEW product_sales_view
WITH DISTRIBUTION DEFAULT NUM_FREQVALUES 50 NUM_QUANTILES 50
SET PROFILE
```
- The following command modifies the previously registered statistics profile. This command also updates the statistics as specified:  

```
RUNSTATS ON VIEW salesdb.product_sales_view
WITH DISTRIBUTION DEFAULT NUM_FREQVALUES 25 NUM_QUANTILES 25
UPDATE PROFILE
```

### Usage notes

- You should run the **RUNSTATS** command in the following cases:
  - On tables that were modified considerably: For example, if many updates were made, if a significant amount of data was inserted or deleted, or if you ran the **LOAD** command without the statistics option during **LOAD**.
  - On tables that were reorganized by using the **REORG** or **REDISTRIBUTE DATABASE PARTITION GROUP** command.
  - On tables that were row compressed.
  - After you create a new index.
  - Before binding applications whose performance is critical.

- When the prefetch quantity is changed.
- On statistical views whose underlying tables that were modified substantially to change the rows that are returned by the view.
- After you run the **LOAD** command with the **STATISTICS** option, if XML columns exist. Use the **RUNSTATS** utility to collect statistics on XML columns because the **LOAD** command does not collect statistics on these columns, even if you use the **STATISTICS** option. When you use the **RUNSTATS** command to collect statistics for XML columns only, statistics for non-XML columns that were collected by the **LOAD** command or a previous execution of the **RUNSTATS** utility are retained. If you previously collected statistics on an XML column, those statistics are either dropped if the current command does not collect statistics on that column or are replaced if the current command does collect statistics on that column.
- The options that you choose depend on the specific table and the application. In general, the following guidelines apply:
  - If the table is a critical table in critical queries, is relatively small, or does not change too much and there is not too much activity on the system itself, it might be worth spending the effort on collecting statistics in as much detail as possible.
  - If the time to collect statistics is limited, if the table is relatively large, or if the table is updated frequently, it might be beneficial to run the **RUNSTATS** command on just the set of columns that are used in predicates. This way, you can run the **RUNSTATS** command more often.
  - If time to collect statistics is limited and the effort to tailor the **RUNSTATS** command on a table-by-table basis is a major issue, consider collecting statistics for the KEY columns only. It is assumed that the index contains the set of columns that are critical to the table and are most likely to appear in predicates.
  - If time to collect statistics is limited and table statistics are to be gathered, consider using the **TABLESAMPLE** option to collect statistics on a subset of the table data.
  - If time to collect statistics is limited and index statistics are to be gathered, consider using the **INDEXSAMPLE** option to collect statistics on a subset of the index data.
  - If skew exists in certain columns and predicates of the type "column = constant", it might be beneficial to specify a larger **NUM\_FREQVALUES** value for the columns.
  - Collect distribution statistics for all columns that you use in equality predicates and for which the distribution of values might be skewed.
  - For columns that have range predicates (for example "column >= constant" or "column BETWEEN constant1 AND constant2") or are of the type "column LIKE '%xyz'", it might be beneficial to specify a larger **NUM\_QUANTILES** value.
  - If storage space is a concern and you do not have much time to collect statistics, do not specify high **NUM\_FREQVALUES** or **NUM\_QUANTILES** values for columns that you do not use in predicates.
  - If you need index statistics, and statistics were never collected on the table that contains the index, statistics on both the table and indexes are calculated.
  - If you do not require statistics for XML columns in the table, you can use the **EXCLUDING XML COLUMNS** parameter to exclude all XML columns. This parameter takes precedence over all other parameters that specify XML columns for statistics collection.
- After the command is run, note the following:

- To release the locks, you must issue a COMMIT.
- To allow new access plans to be generated, you must rebind the packages that reference the target table.
- Running the command on portions of the table can result in inconsistencies as a result of activity on the table since you last issued the command. In this case, a warning message is returned.

If you issue the **RUNSTATS** command on the table only, you might make table-level and index-level statistics inconsistent. For example, you might collect index-level statistics on a table and later delete a significant number of rows from the table. If you then issue the **RUNSTATS** command on the table only, the table cardinality might be less than the value of FIRSTKEYCARD, which is an inconsistency. Similarly, if you collect statistics on a new index when you create it, the table-level statistics might be inconsistent.

- The **RUNSTATS** command drops previously collected distribution statistics if you request table statistics. For example, the **RUNSTATS ON TABLE** and **RUNSTATS ON TABLE ... AND INDEXES ALL** commands cause previously collected distribution statistics to be dropped. If you run the command on indexes only, previously collected distribution statistics are retained. For example, the **RUNSTATS ON TABLE ... FOR INDEXES ALL** command causes the previously collected distribution statistics to be retained. If you run the **RUNSTATS** command on XML columns only, previously collected basic column statistics and distribution statistics are retained. If you previously collected statistics on an XML column, those statistics are either dropped if the current command does not collect statistics on that column or are replaced if the current command does collect statistics on that column.
- When detached partitions exist on a partitioned table, index keys that still belong to detached data partitions that require cleanup are not counted as part of the keys in the statistics. These keys are not counted because they are invisible and no longer part of the table. They will eventually get removed from the index by asynchronous index cleanup. As a result, statistics that are collected before asynchronous index cleanup is run are misleading. If the **RUNSTATS** command is issued before asynchronous index cleanup completes, it might generate a false alarm for index reorganization or index cleanup based on the inaccurate statistics. After asynchronous index cleanup is run, all the index keys that still belong to detached data partitions that require cleanup is removed and this might eliminate the need for index reorganization.

For partitioned tables, you are encouraged to issue the **RUNSTATS** command after an asynchronous index cleanup is completed to generate accurate index statistics in the presence of detached data partitions. To determine whether detached data partitions exist in the table, you can check the status field in the SYSCAT.DATAPARTITIONS catalog view and look for the value L (logically detached), I (index cleanup), or D (detached with dependent MQT).

The **RUNSTATS** command collects statistics for all index partitions of a partitioned index. Statistics in the SYSSTAT.INDEXES view for the partitioned index represent an index partition, except for FIRSTKEYCARD, FIRST2KEYCARD, FIRST3KEYCARD, FIRST4KEYCARD, and FULLKEYCARD statistics. Because these statistics are used in cardinality estimates, they are for the entire index and not for an index partition. Distribution statistics (frequent values and quantiles) are not collected for partitioned indexes, but are gathered if **RUNSTATS** is run on the table. Statistics on the leading columns of a partitioned index might not be as accurate as statistics on the leading columns of a nonpartitioned index.

- When the **RUNSTATS** command is run on a table with an expression-based index, and the **RUNSTATS** command includes that index in the AND INDEXES or FOR INDEXES clause, statistics for expression-based key columns within that index is

also collected and will be associated with the system-generated statistical view associated with the index. Expressions cannot be specified as columns in the **RUNSTATS** command (SQL0205N). To collect customized statistics on expression-based index key columns, you can define a statistics profile on the statistical view that is associated with the index by using the column names as they appear in the statistics view. You also need to define a statistics profile on the base table that includes the index in its **INDEXES** clause.

**Note:** When you define a statistics profile on the expression that is based index's statistical view, an automatically generated statistics profile is also associated with the base table, if one does not exist already. After the statistics profiles are defined, a **RUNSTATS** command on the base table with the **USE PROFILE** clause results in the customized statistics being gathered on the expression-based key columns in the index.

- Distribution statistics are collected on indexes over XML data that is defined on an XML column. If you run the **RUNSTATS** command on a table with the **WITH DISTRIBUTION** parameter, collection of distribution statistics on a column of type XML occurs as follows:
    - The **RUNSTATS** command must collect both distribution statistics and table statistics to collect distribution statistics for indexes over XML data that is defined on an XML column. Table statistics must be gathered in order for distribution statistics to be collected since XML distribution statistics are stored with table statistics.

An index clause is not required to collect XML distribution statistics. Specifying only an index clause does not collect XML distribution statistics. By default, XML distribution statistics use a maximum of 250 quantiles for each index over XML data. When you collect distribution statistics on an XML column, you can change the maximum number of quantiles by specifying a value for the **NUM\_QUANTILES** parameter in the **ON COLUMNS** or the **DEFAULT** clause.

  - Distribution statistics are collected for indexes over XML data of type VARCHAR, DOUBLE, TIMESTAMP, and DATE. Distribution statistics are not collected over indexes of type VARCHAR HASHED.
  - Distribution statistics are not collected for partitioned indexes over XML data that is defined on a partitioned table.
- A special system-generated index in the catalog tables represents the range-ordering property of range-clustered tables. When you collect statistics on this type of table, statistics are also collected for the system-generated index. The statistics reflect the fast access of the range lookups by representing the index as a two-level index with as many pages as the base data table and by clustering the base data perfectly along the index order.
- In the On Dist Cols clause of the command, the Frequency Option and Quantile Option parameters are not supported for column groups. These parameters are supported for single columns.
- Three types of prefetch statistics cannot be computed in DMS mode. In the index statistics in the index catalogs, you will see a -1 value for the following statistics:
  - AVERAGE\_SEQUENCE\_FETCH\_PAGES
  - AVERAGE\_SEQUENCE\_FETCH\_GAP
  - AVERAGE\_RANDOM\_FETCH\_PAGES
- The statistics profile is stored in a visible string format, which represents the **RUNSTATS** command, in the STATISTICS\_PROFILE column of the SYSCAT.TABLES system catalog table.

- Statistics collection on XML-type columns is governed by two Db2 registry variables: **DB2\_XML\_RUNSTATS\_PATHID\_K** and **DB2\_XML\_RUNSTATS\_PATHVALUE\_K**. These two registry variables are similar to the **NUM\_FREQVALUES** parameter in that they specify the number of frequency values to collect. If you do not set the parameters, a default of 200 is used.
- When you start running the **RUNSTATS** command, it acquires an IX table lock on the SYSTABLES table and a U lock on the row of the table on which you are gathering statistics. Operations can still read from the SYSTABLES table, including the row in the table with the U lock. Write operations are also possible if they do not occur against the row with the U lock. However, another reader or writer cannot acquire an S lock on the SYSTABLES table because of the IX lock that the **RUNSTATS** command acquired.
- Statistics are not collected for columns with structured types. If they are specified, columns with these data types are ignored.
- Only AVGCOLLEN and NUMNULLS are collected for columns with LOB or LONG data types. AVGCOLLEN represents the average space in bytes when the column is stored in database memory or a temporary table. This value represents the length of the data descriptor for LOB or LONG data types, except when LOB data is inlined on the data page. The average space that is required to store the column on disk can be different than the value represented by this statistic.
- The **UNSAMPLED DETAILED** option is available to change the way index statistics are collected, but it should be used only in cases where it is clear that the default or **DETAILED** doesn't work.
- When you use the **INDEXSAMPLE** parameter, you cannot specify different index sampling rates for different indexes within a single command. For example, the following command is invalid:  

```
runstats on table orders and index o_ck indexsample system(5),
 index o_ok indexsample system(10)
```

You can use the following two **RUNSTATS** commands to achieve the required result:

```
runstats on table orders and index o_ck indexsample system(5)
runstats on table orders for index o_ok indexsample system(10)
```

- If you modified the table since statistics were last collected on the table or its indexes, you must run **RUNSTATS ON TABLE ... AND INDEXES ALL**. If you use **RUNSTATS ON TABLE ... FOR INDEXES ALL**, the resulting statistics might be inconsistent.

### **SET TAPE POSITION command using the ADMIN\_CMD procedure:**

Sets the positions of tapes for backup and restore operations to streaming tape devices. This command is only supported on Windows operating systems.

#### **Authorization**

One of the following authorities:

- SYSADM
- SYSCtrl
- SYSMAINT

#### **Required connection**

None

## Command syntax

►► SET TAPE POSITION ON *device* TO *position* ◀◀

## Command parameters

### ON *device*

Specifies a valid tape device name. The default value is `\\.\TAPE0`.

### TO *position*

Specifies the mark at which the tape is to be positioned. Db2 for Windows writes a tape mark after every backup image. A value of 1 specifies the first position, 2 specifies the second position, and so on. If the tape is positioned at tape mark 1, for example, archive 2 is positioned to be restored.

## UNQUIESCE DATABASE command using the ADMIN\_CMD procedure:

Restores user access to instances or databases which have been quiesced for maintenance or other reasons. The **UNQUIESCE** command restores user access without necessitating a shutdown and database restart.

## Scope

**UNQUIESCE DB** restores user access to all objects in the quiesced database.

**UNQUIESCE INSTANCE** *instance-name* restores user access to the instance and all databases in the instance *instance-name*.

To stop the instance and unquiesce it and all its databases, issue the **db2stop** command. Stopping and restarting Db2 will unquiesce all instances and databases.

## Authorization

One of the following authorities:

For database level unquiesce:

- SYSADM
- DBADM

For instance level unquiesce:

- SYSADM
- SYSCTRL

## Command syntax

►► UNQUIESCE DB INSTANCE *instance-name* ◀◀

## Required connection

Database

(Database connection is not required for an instance unquiesce.)

## Command parameters

**DB** Unquiesce the database. User access will be restored to all objects in the database.

**INSTANCE** *instance-name*

Access is restored to the instance *instance-name* and all databases in the instance.

## Example : Unquiescing a database

The following command unquiesces the database that had previously been quiesced.

```
db2 unquiesce db
```

## Unquiescing an instance

The following command will unquiesce the instance `instA` that had previously been quiesced.

```
db2 unquiesce instance instA
```

## Usage notes

- In a Db2 pureScale environment, after quiescing a database and restarting the instance, the database will remain quiesced across all members. An explicit **UNQUIESCE DATABASE** command is required to remove the quiesce state.

## UPDATE ALERT CONFIGURATION command using the ADMIN\_CMD procedure:

Updates the alert configuration settings for health indicators.

**Important:** This command or API has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at [http://www.ibm.com/support/knowledgecenter/SSEPGG\\_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html](http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html).

## Authorization

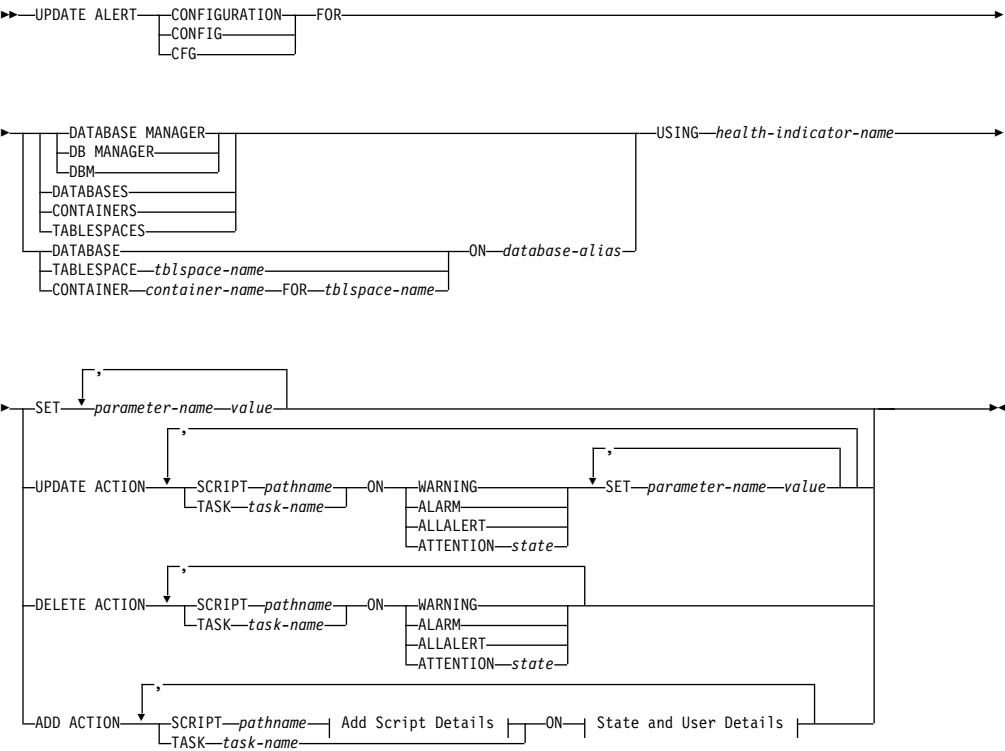
One of the following authorities:

- SYSADM
- SYSMAINT
- SYSCTRL

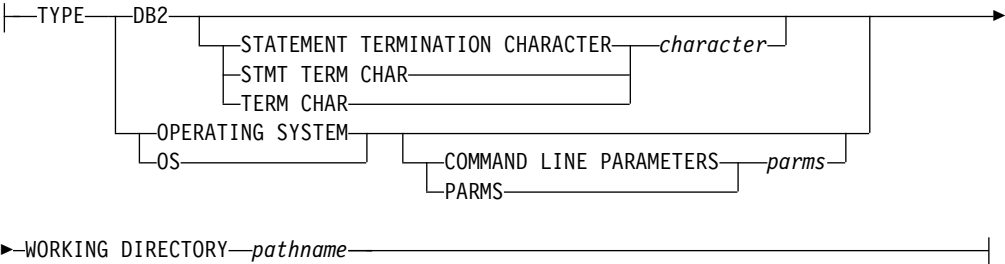
## Required Connection

Instance. An explicit attachment is not required.

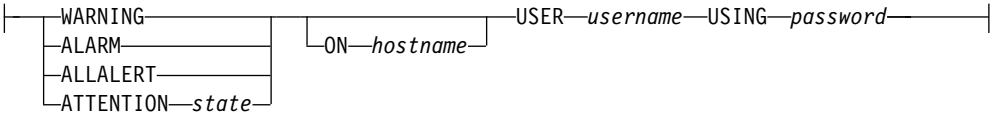
Command Syntax



Add Script Details:



State and User Details:



Command Parameters

DATABASE MANAGER

Updates alert settings for the database manager.

DATABASES

Updates alert settings for all databases managed by the database manager. These are the settings that apply to all databases that do not have custom settings. Custom settings are defined using the **DATABASE ON database-alias** clause.

## CONTAINERS

Updates alert settings for all table space containers managed by the database manager. These are the settings that apply to all table space containers that do not have custom settings. Custom settings are defined using the **CONTAINER** *container-name* **ON** *database-alias* clause.

## TABLESPACES

Updates alert settings for all table spaces managed by the database manager. These are the settings that apply to all table spaces that do not have custom settings. Custom settings are defined using the **TABLESPACE** *tblspace-name* **ON** *database-alias* clause.

## DATABASE **ON** *database-alias*

Updates the alert settings for the database specified using the **ON** *database-alias* clause. If this database has custom settings, then they override the settings for all databases for the instance, which is specified using the **DATABASES** parameter.

## CONTAINER *container-name* **FOR** *tblspace-name* **ON** *database-alias*

Updates the alert settings for the table space container called *container-name*, for the table space specified using the **FOR** *tblspace-name* clause, on the database specified using the **ON** *database-alias* clause. If this table space container has custom settings, then they override the settings for all table space containers for the database, which is specified using the **CONTAINERS** parameter.

## TABLESPACE *tblspace-name* **ON** *database-alias*

Updates the alert settings for the table space called *name*, on the database specified using the **ON** *database-alias* clause. If this table space has custom settings, then they override the settings for all table spaces for the database, which is specified using the **TABLESPACES** parameter.

## USING *health-indicator-name*

Specifies the set of health indicators for which alert configuration will be updated. Health indicator names consist of a two-letter object identifier followed by a name which describes what the indicator measures. For example:

```
db.sort_privmem_util
```

## SET *parameter-name value*

Updates the alert configuration element, *parameter-name*, of the health indicator to the specified value. *parameter-name* must be one of the following values:

- ALARM: the *value* is a health indicator unit.
- WARNING: the *value* is a health indicator unit.
- SENSITIVITY: the *value* is in seconds.
- ACTIONSENABLED: the *value* can be either YES or NO.
- THRESHOLDSCHECKED: the *value* can be either YES or NO.

## UPDATE ACTION SCRIPT *pathname* **ON** [WARNING | ALARM | ALLALERT | ATTENTION

*state*] Specifies that the script attributes of the predefined script with absolute path name *pathname* will be updated according to the following clause:

## SET *parameter-name value*

Updates the script attribute, *parameter-name*, to the specified value. *parameter-name* must be one of the following values:

- SCRIPTTYPE
- OS or DB2 are the valid types.

- WORKINGDIR
- TERMCHAR
- CMDLINEPARMS

The command line parameters that you specify for the operating system script will precede the default supplied parameters. The parameters that are sent to the operating system script are:

- List of user supplied parameters
- Health indicator short name
- Fully qualified object name
- Health indicator value
- Alert state

- USERID
- PASSWORD
- SYSTEM

**UPDATE ACTION TASK *task-name* ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]**

Specifies that the task attributes of the task with name *name* will be updated according to the following clause:

**SET *parameter-name value***

Updates the task attribute, *parameter-name*, to the specified value. *parameter-name* must be one of the following values:

- USERID
- PASSWORD
- SYSTEM

**DELETE ACTION SCRIPT *pathname* ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]**

Removes the action script with absolute path name *pathname* from the list of alert action scripts.

**DELETE ACTION TASK *task-name* ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]**

Removes the action task called *name* from the list of alert action tasks.

**ADD ACTION SCRIPT *pathname* ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]**

Specifies that a new action script with absolute path name *pathname* is to be added, the attributes of which are given by the following:

**TYPE** An action script must be either a Db2 Command script or an operating system script:

- DB2
- OPERATING SYSTEM

If it is a Db2 Command script, then the following clause allows one to optionally specify the character, *character*, that is used in the script to terminate statements:

STATEMENT TERMINATION CHARACTER ;

If it is an operating system script, then the following clause allows one to optionally specify the command-line parameters, *parms*, that would be passed to the script upon invocation: **COMMAND LINE PARAMETERS *parms***

**WORKING DIRECTORY *pathname***

Specifies the absolute path name, *pathname*, of the directory in which the script will be executed.

**USER *username* USING *password***

Specifies the user account, *username*, and associated password, *password*, under which the script will be executed.

**ADD ACTION TASK *name* ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]**

Specifies that a new task, called *name*, is to be added to be run **ON** the specified condition.

**ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]**

Specifies the condition on which the action or task will run. For threshold-based health indicators (HIs), this is **WARNING** or **ALARM**. For state-based HIs, this will be a numeric state as documented for each state-based HI (for example, for the ts.ts\_op\_status health indicator, refer to the **tablespace\_state** monitor element for table space states). **ALLALERTS** handles any changes in the state for threshold-based HIs and state-based HIs (for example, the state changes from warning to normal).

**ATTENTION *state***

Valid numeric values for some of the database health indicator states are given in the following section, as an example for the **ADD ACTION SCRIPT CLP** command option:

- 0 - Active; Normal (ACTIVE)
- 1 - Quiesce pending (QUIESCE\_PEND)
- 2 - Quiesced (QUIESCED)
- 3 - Rollforward (ROLLFWD)

Additional state-based health indicators are defined in the header files sqlmon.h and sqlutil.h.

**Usage notes**

For the **ADD ACTION** option, the supplied *username* and *password* may be exposed in various places where SQL statement text is captured:

- the network (username/password are passed over the wire unencrypted)
- **db2diag** log file
- trace files
- dump file
- snapshot monitor (dynamic SQL snapshot)
- system monitor snapshots
- a number of event monitors (statement, deadlock)
- explain tables
- **db2pd** output (package cache and lock timeout mechanisms, among others)
- Db2 audit records

**UPDATE CONTACT command using the ADMIN\_CMD procedure:**

Updates the attributes of a contact that is defined on the local system. A contact is a user to whom the Scheduler and Health Monitor send messages.

To create a contact, use the **ADD CONTACT** command. The setting of the Database Administration Server (DAS) **contact\_host** configuration parameter determines whether the list is local or global.

## Authorization

None

## Required connection

None. Local execution only: this command cannot be used with a remote connection.

## Command syntax

```
►► UPDATE CONTACT name USING 'keyword' 'value' ►►
```

## Command parameters

### UPDATE CONTACT *name*

The name of the contact that will be updated.

### USING *keyword value*

Specifies the contact parameter to be updated (*keyword*) and the value to which it will be set (*value*). The valid set of keywords is:

#### ADDRESS

The email address that is used by the SMTP server to send the notification.

**TYPE** Whether the address is for an email address or a pager.

#### MAXPAGELEN

The maximum number of characters that the pager can accept.

#### DESCRIPTION

A textual description of the contact. This has a maximum length of 128 characters.

## UPDATE CONTACTGROUP command using the ADMIN\_CMD procedure:

Updates the attributes of a contact group that is defined on the local system. A contact group is a list of users who should be notified by the Scheduler and the Health Monitor.

The setting of the Database Administration Server (DAS) **contact\_host** configuration parameter determines whether the list is local or global.

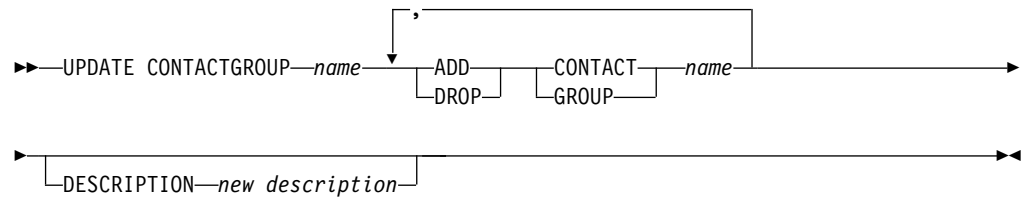
## Authorization

None

## Required Connection

None

## Command Syntax



## Command Parameters

### **CONTACTGROUP** *name*

Name of the contact group which will be updated.

### **ADD CONTACT** *name*

Specifies the name of the new contact to be added to the group. A contact can be defined with the **ADD CONTACT** command after it has been added to a group.

### **DROP CONTACT** *name*

Specifies the name of a contact in the group that will be dropped from the group.

### **ADD GROUP** *name*

Specifies the name of the new contact group to be added to the group.

### **DROP GROUP** *name*

Specifies the name of a contact group that will be dropped from the group.

### **DESCRIPTION** *new description*

Optional. A new textual description for the contact group.

## **UPDATE DATABASE CONFIGURATION command using the ADMIN\_CMD procedure:**

Modifies individual entries in a specific database configuration file. A database configuration file resides on every database partition on which the database has been created.

## Scope

This command updates all database partitions or members by default, except when the following optional clause is specified:

- **MEMBER** to update only one database member for a Db2 pureScale environment, or to update only one database partition in a partitioned database environment.

## Authorization

One of the following authorities:

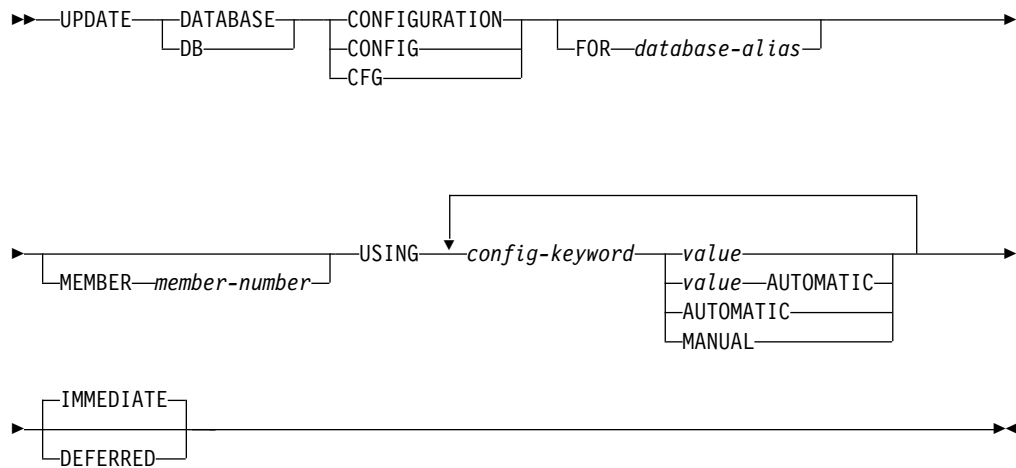
- SYSADM
- SYSCTRL
- SYSMAINT

## Required connection

Instance. An explicit attachment is not required, but a database connection is recommended when the database is active. If the database is listed as remote, an

instance attachment to the remote node is established for the duration of the command. To change a parameter online, you must be connected to the database.

### Command syntax



### Command parameters

#### **FOR database-alias**

Specifies the alias of the database whose configuration is to be updated. Specifying the database alias is not required when a database connection has already been established. You can update the configuration file for another database residing under the same database instance. For example, if you are connected only to database db11, and issue `update db config for alias db22 using .... immediate`:

- If there is no active connection on db22, the update will be successful because only the configuration file needs to be updated. A new connection (which will activate the database) will see the new change in memory.
- If there are active connections on db22 from other applications, the update will work on disk but not in memory. You will receive a warning saying that the database needs to be restarted.

#### **MEMBER member-number**

The **MEMBER** clause specifies to which member the change should be applied. Omission of this clause results in the change being applied to all the members.

#### **USING config-keyword value**

*config-keyword* specifies the database configuration parameter to be updated. *value* specifies the value to be assigned to the parameter.

#### **AUTOMATIC**

Some configuration parameters can be set to **AUTOMATIC**, allowing Db2 database systems to automatically adjust these parameters to reflect the current resource requirements. For a list of configuration parameters that support the **AUTOMATIC** keyword, refer to the configuration parameters summary. If a value is specified along with the **AUTOMATIC** keyword, it might influence the automatic calculations. For specific details about this behavior, refer to the documentation for the configuration parameter.

**MANUAL** Disables automatic tuning for the configuration parameter. The parameter is set to its current internal value and is no longer updated automatically.

**IMMEDIATE**

Make the changes immediately, while the database is running. **IMMEDIATE** is the default action, but it requires a database connection to be effective.

This is a default clause when operating in the CLPPlus interface as well. **IMMEDIATE** need not be called when using CLPPlus processor.

**DEFERRED**

Make the changes only in the configuration file, so that the changes take effect the next time you reactivate the database.

## Examples

### Update database configuration on a multi-partition instance

This example demonstrates how to update database configuration parameter **MAXAPPLS** from 10 to 50 for a database named **SAMPLE**.

A user has a multi-partition instance that has 4 partitions as defined in the `db2nodes.cfg`:

```
10 gilera 0
20 gilera 1
30 motobi 0
40 motobi 1
```

The user has created the **SAMPLE** database on the instance. The catalog partition for **SAMPLE** is on member 10. Let us assume the user is logged on to system `motobi`.

Since the default behavior for a multi-partition instance is to update the database configurations on all database partitions, the following command issued by users will result in the same value for **MAXAPPLS** across all database partitions:

```
db2 update db cfg for sample using maxappls 50
```

To update **MAXAPPLS** only on member 30, the following commands may be issued:

```
db2 update db cfg for sample member 30 using maxappls 50
```

### Usage notes

To view or print a list of the database configuration parameters, use the **GET DATABASE CONFIGURATION** command.

To reset all the database configuration parameters to the recommended defaults, use the **RESET DATABASE CONFIGURATION** command.

To change a database configuration parameter, use the **UPDATE DATABASE CONFIGURATION** command. For example, to change the logging mode to “archival logging” on a single-partition database environment containing a database called **ZELLMART**, use:

```
db2 update db cfg for zellmart using logarchmeth1 logretain
```

To check that the **logarchmeth1** configuration parameter has changed, use:

```
db2 get db cfg for zellmart
```

For example, to change the logging mode to “archival logging” on all partitions (provided the registry variable **DB2\_UPDDBCFG\_SINGLE\_DBPARTITION** is set, by default, to NULL or FALSE) in a multiple-partitioned database environment containing a database called “zellmart”, use:

```
db2 update db cfg for zellmart using logarchmeth1 logretain
```

To check that the **logarchmeth1** configuration parameter has changed on all database partitions, use:

```
db2_all ";db2 get db cfg for zellmart"
```

Using the previous example, but to update the logging mode to only one specific partition (30), use:

```
db2 update db cfg for zellmart member 30 using logarchmeth1 logretain
```

Optionally, you can leverage the SYSIBMADM.DBCFG view to get data from all partitions without having to use **db2\_all**.

If you are working on a UNIX operating system, and you have the **grep** command, you can use the following command to view only the **logarchmeth1** values:

```
db2_all ";db2 get db cfg for zellmart | grep -i logarchmeth1"
```

For more information about Db2 database configuration parameters and the values available for each type of database node, see the individual configuration parameter descriptions. The values of these parameters differ for each type of database node configured (server, client, or server with remote clients).

Not all parameters can be updated.

Some changes to the database configuration file become effective only after they are loaded into memory. All applications must disconnect from the database before this can occur. For more information about which parameters are configurable online and which ones are not, see summary list of configuration parameters.

For example, to change the **sortheap** database configuration parameter online for the SALES database, enter the following commands:

```
db2 connect to sales
db2 update db cfg using sortheap 1000
db2 connect reset
```

If an error occurs, the database configuration file does not change. The database configuration file cannot be updated if the checksum is invalid. This might occur if the database configuration file is changed without using the appropriate command. If this happens, the database must be restored to reset the database configuration file.

## Compatibilities

For compatibility with previous versions:

- **DBPARTITIONNUM** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.

## UPDATE DATABASE MANAGER CONFIGURATION command using the ADMIN\_CMD procedure:

Modifies individual entries in the database manager configuration file.

## Authorization

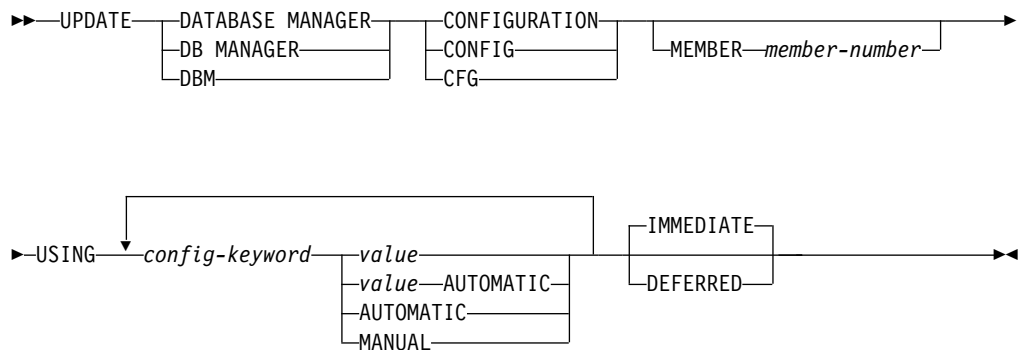
One of the following:

- SYSADM
- Member of Local Administrator Group on Windows operating system
- Instance Owner on UNIX or Linux operating system

## Required connection

None or instance. An instance attachment is not required to perform local DBM configuration operations, but is required to perform remote DBM configuration operations. To update the database manager configuration for a remote instance, it is necessary to first attach to that instance. To update a configuration parameter online, it is also necessary to first attach to the instance.

## Command syntax



## Command parameters

### **MEMBER** *member-number*

The **MEMBER** clause specifies to which member the change should be applied. Omission of this clause results in the change being applied to all the members.

### **USING** *config-keyword value*

Specifies the database manager configuration parameter to be updated. For a list of configuration parameters, refer to the configuration parameters summary. *value* specifies the value to be assigned to the parameter.

### **AUTOMATIC**

Some configuration parameters can be set to **AUTOMATIC**, allowing Db2 to automatically adjust these parameters to reflect the current resource requirements. For a list of configuration parameters that support the **AUTOMATIC** keyword, refer to the configuration parameters summary. If a value is specified along with the **AUTOMATIC** keyword, it might influence the automatic calculations. For specific details about this behavior, refer to the documentation for the configuration parameter.

### **Note:**

**MANUAL** Disables automatic tuning for the configuration parameter. The parameter is set to its current internal value and is no longer updated automatically.

## **IMMEDIATE**

Make the changes right now, dynamically, while the instance is running. **IMMEDIATE** is the default, but it requires an instance attachment to be effective.

## **DEFERRED**

Make the changes only in the configuration file, so that the changes take effect when the instance is restarted.

This is a default clause when operating in the CLPPlus interface. **DEFERRED** need not be called when using CLPPlus processor.

## **Usage notes**

To view or print a list of the database manager configuration parameters, use the **GET DATABASE MANAGER CONFIGURATION** command. To reset the database manager configuration parameters to the recommended database manager defaults, use the **RESET DATABASE MANAGER CONFIGURATION** command. For more information about database manager configuration parameters and the values of these parameters appropriate for each type of database node configured (server, client, or server with remote clients), see individual configuration parameter descriptions.

Not all parameters can be updated.

Some changes to the database manager configuration file become effective only after they are loaded into memory. For more information about which parameters are configurable online and which ones are not, see the configuration parameter summary. Server configuration parameters that are not reset immediately are reset during execution of **db2start**. For a client configuration parameter, parameters are reset the next time you restart the application. If the client is the command line processor, it is necessary to invoke **TERMINATE**.

For example, to change the **DIAGLEVEL** database manager configuration parameter online for the eastern instance of the database manager, enter the following command:

```
db2 attach to eastern
db2 update dbm cfg using DIAGLEVEL 1
db2 detach
```

If an error occurs, the database manager configuration file does not change.

The database manager configuration file cannot be updated if the checksum is invalid. This can occur if you edit database manager configuration file and do not use the appropriate command. If the checksum is invalid, you must reinstall the database manager to reset the database manager configuration file.

When you update the **SVCENAME**, or **TPNAME** database manager configuration parameters for the current instance, if LDAP support is enabled and there is an LDAP server registered for this instance, the LDAP server is updated with the new value or values.

**UPDATE HEALTH NOTIFICATION CONTACT LIST** command using the **ADMIN\_CMD** procedure:

Updates the contact list for notification about health alerts issued by an instance.

## Authorization

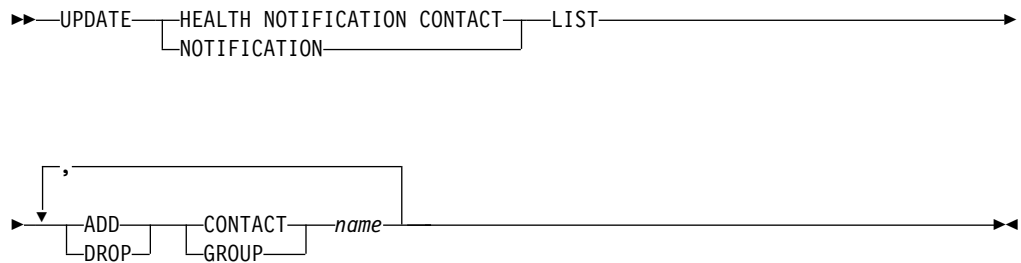
One of the following authorities:

- SYSADM
- SYSCtrl
- SYSMAINT

## Required Connection

Instance. An explicit attachment is not required.

## Command Syntax



## Command Parameters

### **ADD GROUP** *name*

Add a new contact group that will notified of the health of the instance.

### **ADD CONTACT** *name*

Add a new contact that will notified of the health of the instance.

### **DROP GROUP** *name*

Removes the contact group from the list of contacts that will notified of the health of the instance.

### **DROP CONTACT** *name*

Removes the contact from the list of contacts that will notified of the health of the instance.

## **UPDATE HISTORY** command using the **ADMIN\_CMD** procedure:

Updates the location, device type, comment, or status in a database history records entry.

## Authorization

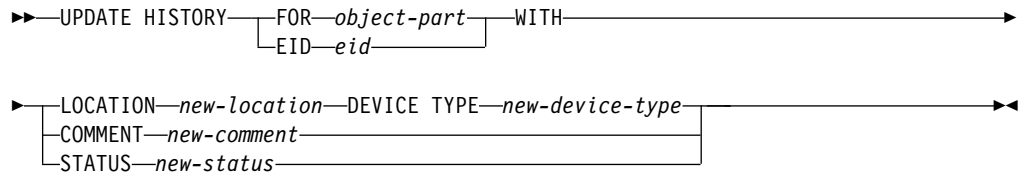
One of the following authorities:

- SYSADM
- SYSCtrl
- SYSMAINT
- DBADM

## Required connection

Database

## Command syntax



## Command parameters

### FOR *object-part*

Specifies the identifier for the history entry to be updated. It is a time stamp with an optional sequence number from 001 to 999. This parameter cannot be used to update the entry status. To update the entry status, specify an EID instead.

### EID *eid*

Specifies the history entry ID.

### LOCATION *new-location*

Specifies the new physical location of a backup image. The interpretation of this parameter depends on the device type.

### DEVICE TYPE *new-device-type*

Specifies a new device type for storing the backup image. Valid device types are:

<b>D</b>	Disk
<b>K</b>	Diskette
<b>T</b>	Tape
<b>A</b>	Tivoli Storage Manager
<b>F</b>	Snapshot backup
<b>f</b>	Snapshot backup that was generated by a custom script.
<b>U</b>	User exit
<b>P</b>	Pipe
<b>N</b>	Null device
<b>X</b>	XBSA
<b>Q</b>	SQL statement
<b>O</b>	Other

### COMMENT *new-comment*

Specifies a new comment to describe the entry. The size of *new-comment* cannot exceed 30 ASCII characters.

### STATUS *new-status*

Specifies a new status for an entry. Only backup entries can have their status updated. Valid values are:

<b>A</b>	Active. The backup image is on the active log chain. Most entries are active.
----------	-------------------------------------------------------------------------------

- I** Inactive. Backup images that no longer correspond to the current log sequence, also called the current log chain, are flagged as inactive.
- E** Expired. Backup images that are no longer required, because there are more than NUM\_DB\_BACKUPS active images, are flagged as expired.
- D** Deleted. Backup images that are no longer available for recovery should be marked as having been deleted.
- X** Do not delete. Recovery database history records file entries that are marked DB2HISTORY\_STATUS\_DO\_NOT\_DELETE will not be pruned by calls to the **PRUNE HISTORY** command, running the ADMIN\_CMD procedure with **PRUNE HISTORY**, calls to the db2Prune API, or automated recovery database history records pruning. You can use the DB2HISTORY\_STATUS\_DO\_NOT\_DELETE status to protect key recovery file entries from being pruned and the recovery objects associated with them from being deleted. Only log files, backup images, and load copy images can be marked as DB2HISTORY\_STATUS\_DO\_NOT\_DELETE.

### Example

Let's say there was a full database backed up on April 13, 1997 at 10:00 a.m. and backed up to 2 target devices. The object-part of database history records entries for resulting backup images should be 19970413100000001 and 19970413100000002, respectively. The first 14 digits (19970413100000) of object-part is the time stamp associated with this backup image and the last 3 digits (001 or 002) is the sequence number of this backup image. To update database history records entries for this full database backup, please enter:

```
db2 update history for 19970413100000001 with
 location /backup/dbbackup.1 device type D
db2 update history for 19970413100000002 with location /backup/dbbackup.2 device type D
```

### Usage notes

The primary purpose of the database history records is to record information, but the data contained in the history is used directly by automatic restore operations. During any restore where the **AUTOMATIC** option is specified, the history of backup images and their locations will be referenced and used by the restore utility to fulfill the automatic restore request. If the automatic restore function is to be used and backup images have been relocated since they were created, it is recommended that the database history record for those images be updated to reflect the current location. If the backup image location in the database history is not updated, automatic restore will not be able to locate the backup images, but manual restore commands can still be used successfully.

### UPDATE STMM TUNING command by using the ADMIN\_CMD procedure:

Update the user preferred database member number on which the self-tuning memory manager (STMM) tuner is created.

### Authorization

The privileges that are held by the authorization ID of the statement must include at least one of the following authorities:

- DBADM
- DATAACCESS
- SQLADM

### Required connection

Database

### Command syntax

►►—UPDATE—STMM—TUNING—MEMBER—*member-number*—◀◀

### Command parameter

#### *member-number*

The value of *member-number* is an integer.

In a partitioned database environment:

- If a valid member number is specified, the database server runs the STMM memory tuner on that member.
- If -1 or a nonexistent member number is specified, the database server selects an appropriate member on which to run the STMM memory tuner.

In a Db2 pureScale environment:

- If a valid member number is specified, database server runs the STMM memory tuner on that member.
- If -2 is specified, then the database server enables STMM tuners to run and tune independently on each member.
- If -1 or a nonexistent member number is specified, the database server selects an appropriate member on which to run the STMM memory tuner.

### Example

In a partitioned database environment, update the user preferred self-tuning memory manager (STMM) tuning database partition to member 3.

```
CALL SYSPROC.ADMIN_CMD('update stmm tuning member 3')
```

### Usage notes

- The STMM tuning process periodically checks for a change in the user preferred STMM tuning member number value. The STMM tuning process moves to the user preferred STMM tuning member if *member-number* exists and is an active member. If the specified STMM tuning member does not exist, the database server selects an appropriate member. After this command changes the STMM tuning member number, an immediate change is made to the current STMM tuning member number.
- Command execution status is returned in the SQLCA resulting from the **CALL** statement.
- This command commits its changes in the **ADMIN\_CMD** procedure.

### Compatibilities

For compatibility with previous versions:

- **DBPARTITIONNUM** can be substituted for **MEMBER**, except when the **DB2\_ENFORCE\_MEMBER\_SYNTAX** registry variable is set to ON.

## **ADMIN\_COPY\_SCHEMA procedure - Copy a specific schema and its objects**

The **ADMIN\_COPY\_SCHEMA** procedure is used to copy a specific schema and all objects contained in it.

The new target schema objects will be created using the same object names as the objects in the source schema, but with the target schema qualifier.

The **ADMIN\_COPY\_SCHEMA** procedure can be used to copy tables with or without the data of the original tables.

### **Authorization**

In order for the schema copy to be successful, the user must have the **CREATE\_SCHEMA** privilege as well as object-specific privileges.

*Example:* **CREATE\_TABLE** privilege is needed to copy a table and **CREATE\_INDEX** privilege is needed to copy an index under the **ADMIN\_COPY\_SCHEMA** command.

If a table in the source schema is protected by label based access control (LBAC), the user ID must have LBAC credentials that allow creating that same protection on the target table. If copying with data, the user ID must also have LBAC credentials that allow both reading the data from the source table and writing that data to the target table.

**EXECUTE** privilege on the **ADMIN\_COPY\_SCHEMA** procedure is also needed.

### **Default PUBLIC privilege**

In a non-restrictive database, **EXECUTE** privilege is granted to **PUBLIC** when the procedure is automatically created.

### **Syntax**

```
►► ADMIN_COPY_SCHEMA (—sourceschema—, —targetschema—, —copymode—, —————→
►—objectowner—, —sourcetbsp—, —targettbsp—, —errortabschema—, —errortab—) ►◄
```

The schema is **SYSPROC**.

### **Procedure parameters**

#### *sourceschema*

An input argument of type **VARCHAR(128)** that specifies the name of the schema whose objects are being copied. The name is case-sensitive.

#### *targetschema*

An input argument of type **VARCHAR(128)** that specifies a unique schema name to create the copied objects into. The name is case-sensitive. If the schema name already exists, the procedure call will fail and return a message indicating that the schema must be removed before invoking the procedure.

### *copymode*

An input argument of type VARCHAR(128) that specifies the mode of copy operation. Valid options are:

- 'DDL': create empty copies of all supported objects from the source schema.
- 'COPY': create empty copies of all objects from the source schema, then load each target schema table with data. Load is done in 'NONRECOVERABLE' mode. A backup must be taken after calling the ADMIN\_COPY\_SCHEMA, otherwise the copied tables will be inaccessible following recovery.
- 'COPYNO': create empty copies of all objects from the source schema, then load each target schema table with data. Load is done in 'COPYNO' mode.

**Note:** If *copymode* is 'COPY' or 'COPYNO', a fully qualified filename, for example 'COPYNO /home/mckeough/loadoutput', can be specified along with the *copymode* parameter value. When a path is passed in, load messages will be logged to the file indicated. The file name must be writable by the user ID used for fenced routine invocations on the instance. If no path is specified, then load message files will be discarded (default behavior).

### *objectowner*

An input argument of type VARCHAR(128) that specifies the authorization ID to be used as the owner of the copied objects. If NULL, then the owner will be the authorization ID of the user performing the copy operation.

### *sourcetbsp*

An input argument of type CLOB(2 M) that specifies a list of source table spaces for the copy, separated by commas. Delimited table space names are supported. For each table being created, any table space found in this list, and the tables definition, will be converted to the nth entry in the *targettbsp* list. If NULL is specified for this parameter, new objects will be created using the same table spaces as the source objects use.

### *targettbsp*

An input argument of type CLOB(2 M) that specifies a list of target table spaces for the copy, separated by commas. Delimited table space names are supported. One table space must be specified for each entry in the *sourcetbsp* list of table spaces. The nth table space in the *sourcetbsp* list will be mapped to the nth table space in the *targettbsp* list during DDL replay. It is possible to specify 'SYS\_ANY' as the final table space (an additional table space name, that does not correspond to any name in the source list). When 'SYS\_ANY' is encountered, the default table space selection algorithm will be used when creating objects (refer to the IN *tablespace-name1* option of the CREATE TABLE statement documentation for further information about the selection algorithm). If NULL is specified for this parameter, new objects will be created using the same table spaces as the source objects use.

### *errortabschema*

An input and output argument of type VARCHAR(128) that specifies the schema name of a table containing error information for objects that could not be copied. This table is created for the user by the ADMIN\_COPY\_SCHEMA procedure in the SYSTOOLSPACE table space. If no errors occurred, then this parameter is NULL on output.

### *errortab*

An input and output argument of type VARCHAR(128) that specifies the name of a table containing error information for objects that could not be copied. This table is created for the user by the ADMIN\_COPY\_SCHEMA procedure in the SYSTOOLSPACE table space. This table is owned by the user ID that invoked the procedure. If no errors occurred, then this parameter is NULL on

output. If the table cannot be created or already exists, the procedure operation fails and an error message is returned. The table must be cleaned up by the user following any call to the ADMIN\_COPY\_SCHEMA procedure; that is, the table must be dropped in order to reclaim the space it is consuming in SYSTOOLSPACE.

Table 29. ADMIN\_COPY\_SCHEMA errortab format

Column name	Data type	Description
OBJECT_SCHEMA	VARCHAR(128)	object_schema - Object schema monitor element
OBJECT_NAME	VARCHAR(128)	object_name - Object name monitor element
OBJECT_TYPE	VARCHAR(30)	objtype - Object type monitor element
SQLCODE	INTEGER	The error SQLCODE.
SQLSTATE	CHAR(5)	The error SQLSTATE.
ERROR_TIMESTAMP	TIMESTAMP	Time of failure for the operation that failed.
STATEMENT	CLOB(2 M)	DDL for the failing object. If the failure occurred when data was being loaded into a target table, this field contains text corresponding to the load command that failed.
DIAGTEXT	CLOB(2 K)	Error message text for the failed operation.

## Restrictions

- Only DDL *copymode* is supported for HADR databases.
- XML with COPY or COPY NO is not supported.
- Using the ADMIN\_COPY\_SCHEMA procedure with the COPYNO option places the table spaces in which the target database object resides in backup pending state. After the load operation completes, target schema tables are in set integrity pending state, and the ADMIN\_COPY\_SCHEMA procedure issues a SET INTEGRITY statement to get the tables out of this state. Because the table spaces are already in backup pending state, the SET INTEGRITY statement fails. For information about how to resolve this problem, see “Copying a schema”.

## Usage notes

- References to fully qualified objects within the objects being copied will not be modified. The ADMIN\_COPY\_SCHEMA procedure only changes the qualifying schema of the object being created, not any schema names that appear within SQL expressions for those objects. This includes objects such as generated columns and trigger bodies.
- This procedure does not support copying the following objects:
  - index extensions
  - nicknames
  - packages
  - typed tables
  - array types

- user-defined structured types (and their transform functions)
- typed views
- jars (Java routine archives)
- staging tables
- aliases with base objects that do not belong to the same source schema
- partitioned tables that require table space mapping
- If one of these objects exists in the schema being copied, the object is not copied but an entry is added to the error table indicating that the object has not been copied.
- When a replicated table is copied, the new copy of the table does not have subscriptions enabled. The table is re-created as a basic table only.
- The operation of this procedure requires the existence of the SYSTOOLSPACE table space. This table space is used to hold metadata used by the ADMIN\_COPY\_SCHEMA procedure as well as error tables returned by this procedure. If the table space does not exist, an error is returned.
- Statistics for the objects in the target schema are set to default.
- If a table has a generated identity column, and *copymode* is either 'COPY' or 'COPYNO', the data values from the source table are preserved during the load.
- A new catalog entry is created for each external routine, referencing the binary of the original source routine.
- If a table is in set integrity pending state at the beginning of the copy operation, the data is not loaded into the target table and an entry is logged in *errortab* indicating that the data was not loaded for that table.
- If a Load or DDL operation fails, an entry is logged in *errortab* for any object that was not created. All objects that are successfully created remain. To recover, a manual load can be initiated, or the new schema can be dropped using the ADMIN\_DROP\_SCHEMA procedure and the ADMIN\_COPY\_SCHEMA procedure can be called again.
- During DDL replay, the default schema is overridden to the target schema if it matches the source schema.
- The function path used to compile a trigger, view or SQL function is the path used to create the source object, with the following exception: if the object's function path contains the source schema name, this entry in the path is modified to the target schema name during DDL replay.
- Running multiple ADMIN\_COPY\_SCHEMA procedures will result in deadlocks. Only one ADMIN\_COPY\_SCHEMA procedure call should be issued at a time. Changes to tables in the source schema during copy processing might mean that the data in the target schema is not identical following a copy operation.
- Careful consideration should be taken when copying a schema with tables from a table space in a single-partition database partition group to a table space in a multiple-partition database partition group. Unless automatic distribution key selection is preferred, the distribution key should be defined on the tables before the copy schema operation is undertaken. Altering the distribution key can only be done to a table whose table space is associated with a single-partition database partition group.

### Transactional considerations

- If the ADMIN\_COPY\_SCHEMA procedure is forced to roll back due to a deadlock or lock timeout during its processing, any work performed in the unit of work that called the ADMIN\_COPY\_SCHEMA procedure is also rolled back.

- If a failure occurs during the DDL phase of the copy, all the changes that were made to the target schema are rolled back to a savepoint.
- If *copymode* is set to 'COPY' or 'COPYNO', the ADMIN\_COPY\_SCHEMA procedure commits once the DDL phase of the copy is complete, also committing any work done in the unit of work that called the procedure.

### Example

```
CALL SYSPROC.ADMIN_COPY_SCHEMA('SOURCE_SCHEMA', 'TARGET_SCHEMA',
 'COPY', NULL, 'SOURCETS1', SOURCETS2', 'TARGETTS1, TARGETTS2,
 SYS_ANY', 'ERRORSCHEMA', 'ERRORNAME')
```

## ADMIN\_DROP\_SCHEMA procedure - Drop a specific schema and its objects

The ADMIN\_DROP\_SCHEMA procedure is used to drop a specific schema and all objects contained in it.

### Syntax

```
►► ADMIN_DROP_SCHEMA(—schema—, —dropmode—, —errortabschema—, —errortab—)
```

The schema is SYSPROC.

### Procedure parameters

#### *schema*

An input argument of type VARCHAR(128) that specifies the name of the schema being dropped. The name must be specified in uppercase characters.

#### *dropmode*

Reserved for future use and should be set to NULL.

#### *errortabschema*

An input and output argument of type VARCHAR(128) that specifies the schema name of a table containing error information for objects that could not be dropped. The name is case-sensitive. This table is created for the user by the ADMIN\_DROP\_SCHEMA procedure in the SYSTOOLSPACE table space. If no errors occurred, then this parameter is NULL on output.

#### *errortab*

An input and output argument of type VARCHAR(128) that specifies the name of a table containing error information for objects that could not be dropped. The name is case-sensitive. This table is created for the user by the ADMIN\_DROP\_SCHEMA procedure in the SYSTOOLSPACE table space. This table is owned by the user ID that invoked the procedure. If no errors occurred, then this parameter is NULL on output. If the table cannot be created or already exists, the procedure operation fails and an error message is returned. The table must be cleaned up by the user following any call to ADMIN\_DROP\_SCHEMA; that is, the table must be dropped in order to reclaim the space it is consuming in SYSTOOLSPACE.

Table 30. ADMIN\_DROP\_SCHEMA errortab format

Column name	Data type	Description
OBJECT_SCHEMA	VARCHAR(128)	object_schema - Object schema monitor element

Table 30. ADMIN\_DROP\_SCHEMA errortab format (continued)

Column name	Data type	Description
OBJECT_NAME	VARCHAR(128)	object_name - Object name monitor element
OBJECT_TYPE	VARCHAR(30)	objtype - Object type monitor element
SQLCODE	INTEGER	The error SQLCODE.
SQLSTATE	CHAR(5)	The error SQLSTATE.
ERROR_TIMESTAMP	TIMESTAMP	Time that the drop command failed.
STATEMENT	CLOB(2 M)	DDL for the failing object.
DIAGTEXT	CLOB(2 K)	Error message text for the failed drop command.

## Authorization

One of the following authorizations is required:

- EXECUTE privilege on the ADMIN\_DROP\_SCHEMA procedure
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, drop authority is needed on all objects being removed for the user calling this procedure.

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

## Example

```
CALL SYSPROC.ADMIN_DROP_SCHEMA('SCHNAME', NULL, 'ERRORSCHEMA', 'ERRORTABLE')
```

The following is an example of output for this procedure.

Value of output parameters

-----

Parameter Name : ERRORTABSCHEMA

Parameter Value : ERRORSCHEMA <-- error!

Parameter Name : ERRORTAB

Parameter Value : ERRORTABLE <-- error!

Return Status = 0

The return status is not zero only when an internal error has been detected (for example, if SYSTOOLSPACE does not exist).

Errors can be checked by querying the error table:

```
SELECT * FROM ERRORSCHEMA.ERRORTABLE
```

## Usage notes

- If objects in another schema depend on an object being dropped, the default DROP statement semantics apply.

- This procedure does not support dropping the following objects:
  - Index extensions
  - Nicknames
  - Packages
  - Typed tables
  - Array types
  - User-defined structured types (and their transform functions)
  - Typed views
  - Jars (Java routine archives)
  - Staging tables
  - XSR objects
- If one of these objects exists in the schema being dropped, neither the object nor the schema is dropped, and an entry is added to the error table indicating that the object was not dropped.
- The operation of this procedure requires the existence of the SYSTOOLSPACE table space. This table space is used to hold metadata used by the ADMIN\_DROP\_SCHEMA procedure as well as error tables returned by this procedure. If the table space does not exist, an error is returned.

### **ADMIN\_EST\_INLINE\_LENGTH function - Estimate length required to inline data**

The ADMIN\_EST\_INLINE\_LENGTH function returns an estimate of the inline length that is required to inline the data stored in an XML column, BLOB column, CLOB column, or DBCLOB column.

If the data cannot be inlined, the function returns a negative value.

If the data is already inlined, the function returns the actual length of the inlined data.

### **Syntax**

►► ADMIN\_EST\_INLINE\_LENGTH(—*column-name*—)—————►◄

The schema is SYSIBM.

### **Return value**

This function returns either an INTEGER value that represents the estimated inline length (in bytes) of the data, or one of the following values:

**NULL** The inputs are NULL.

- 1 The data cannot be inlined because there is no valid inline length that would allow the column value to be inlined.
- 2 The estimated inline length of the document cannot be determined because the document was inserted and stored using an earlier product release.

### **Function parameters**

#### ***column-name***

Identifies a column of the base table with a data type of XML, BLOB, CLOB, or

DBCLOB (SQLSTATE 42884). The column must directly or indirectly reference the column of a base table that is not generated based on an expression (SQLSTATE 42815).

## Example

*Example 1:* The following example returns the estimated inline length of three XML documents that are contained in XML column xml\_doc1 of TAB1 table.

```
db2 => SELECT PK, ADMIN_IS_INLINED(xml_doc1) as IS_INLINED,
 ADMIN_EST_INLINE_LENGTH(xml_doc1) as EST_INLINE_LENGTH
 from TAB1
```

This query results in the following output:

PK	IS_INLINED	EST_INLINE_LENGTH
1	1	292
2	0	450
3	0	454

3 record(s) selected.

In the example, the ADMIN\_IS\_INLINED function indicates that the first document is inlined. Therefore, the ADMIN\_EST\_INLINE\_LENGTH function returns the actual length of the inlined XML document. The second document is not inlined, so the ADMIN\_EST\_INLINE\_LENGTH function returns the estimated inline length that is required to inline the second XML document.

*Example 2:* The following example returns the estimated inline length of one XML document that is contained in the XML column xml\_doc1 of the TAB1 table. This example includes a predicate.

```
db2 => SELECT PK, ADMIN_IS_INLINED(xml_doc1) as IS_INLINED,
 ADMIN_EST_INLINE_LENGTH(xml_doc1) as EST_INLINE_LENGTH
 from TAB1 where PK=2
```

This query results in the following output:

PK	IS_INLINED	EST_INLINE_LENGTH
2	0	450

1 record(s) selected.

*Example 3:* The following example returns the estimated inline length of three CLOB data that are contained in CLOB column clob\_1 of the TAB1 table.

```
db2 => SELECT PK, ADMIN_IS_INLINED(clob_1) as IS_INLINED,
 ADMIN_EST_INLINE_LENGTH(clob_1) as EST_INLINE_LENGTH
 from TAB1
```

This query results in the following output:

PK	IS_INLINED	EST_INLINE_LENGTH
1	1	68
2	0	3665
3	0	-1

3 record(s) selected.

## Usage notes

- XML columns are supported only when the XML documents were inserted using Db2 Version 9.7 or later. XML documents inserted before this release have a different storage format. When the ADMIN\_EST\_INLINE\_LENGTH function encounters an incorrect storage format, it returns a value of -2.
- If you plan to increase the column inline length, remember that this length cannot be reduced.
- Increasing the inline length also increases the total row size and might affect the performance of buffer pools. The total row size has the following limits.

Table 31. Row size limits

Page size	Row size limit	Inline length limit
4K	4005	4001
8K	8101	8097
16K	16 293	16 289
32K	32 677	32 673

- The estimated inline length might not be accurate if the XML storage object page size is not same as the base table page size.

## ADMIN\_GET\_ENCRYPTION\_INFO - Get database encryption settings

The ADMIN\_GET\_ENCRYPTION\_INFO table function returns the current encryption settings for a database.

### Authorization

The following authorization is required:

- EXECUTE privilege on the routine. EXECUTE privilege on this routine can be granted only by the security administrator (SECADM).

### Default PUBLIC privilege

None

### Syntax

►►—ADMIN\_GET\_ENCRYPTION\_INFO—(—)—————►►

The schema is SYSPROC.

### Information returned

Table 32. Information returned by the ADMIN\_GET\_ENCRYPTION\_INFO routine

Column name	Data type	Description
OBJECT_NAME	VARCHAR(128)	Name of the object being encrypted.
OBJECT_TYPE	VARCHAR(20)	Type of object being encrypted.
ALGORITHM	VARCHAR(16)	Encryption algorithm used.
ALGORITHM_MODE	VARCHAR(16)	Encryption mode used.
KEY_LENGTH	SMALLINT	Encryption key length.

Table 32. Information returned by the ADMIN\_GET\_ENCRYPTION\_INFO routine (continued)

Column name	Data type	Description
MASTER_KEY_LABEL	VARCHAR(255)	Master key label associated with the master key used.
KEYSTORE_NAME	VARCHAR(255)	Absolute path of the keystore file location.
KEYSTORE_TYPE	VARCHAR(20)	Type of keystore.
KEYSTORE_HOST	VARCHAR(255)	Host name of the server where the keystore file is located.  When the database instance is configured for PKCS #11 keystore, the KEYSTORE_HOST column contains a semi-colon separated list of information about the PKCS #11 keystore. The format is:  <slot label> ;<token manufacturer> ; <token model> ;<serial number> ;
KEYSTORE_IP	VARCHAR(40)	IP address of the server where the keystore file is located.
KEYSTORE_IP_TYPE	VARCHAR(16)	Type of the IP address of the keystore (IPV4 or IPV6).
PREVIOUS_MASTER_KEY_LABEL	VARCHAR(255)	Master key label before the last master key rotation took place. If a master key rotation has not occurred, this value is the master key label with which the database is created.
AUTH_ID	VARCHAR(128)	Authorization ID that was used during the last master key rotation.
APPL_ID	VARCHAR(128)	Application ID that was used during the last master key rotation.
ROTATION_TIME	TIMESTAMP	Timestamp when the last master key rotation took place.

## Examples

1. Retrieve the current database encryption settings.

```
SELECT OBJECT_NAME, OBJECT_TYPE, ALGORITHM, ALGORITHM_MODE, KEY_LENGTH, MASTER_KEY_LABEL
FROM TABLE(SYSPROC.ADMIN_GET_ENCRYPTION_INFO())
```

This query returns the following output:

OBJECT_NAME	OBJECT_TYPE	ALGORITHM	...
-----	-----	-----	...
SAMPLE	DATABASE	AES	...

Query output (continued):

...	ALGORITHM_MODE	KEY_LENGTH	MASTER_KEY_LABEL
...	-----	-----	-----
...	CBC	256	DB2_SYSGEN_geoffrey_SAMPLE_2013-12-11-09.23.56

2. Retrieve information about the instance keystore setting.

```
SELECT KEYSTORE_NAME, KEYSTORE_TYPE, KEYSTORE_HOST, KEYSTORE_IP, KEYSTORE_IP_TYPE
FROM TABLE(SYSPROC.ADMIN_GET_ENCRYPTION_INFO())
```

This query returns the following output:

KEYSTORE_NAME	KEYSTORE_TYPE	...
-----	-----	...
/home/hotel70/geoffrey/sql11b/keystore.p12	PKCS12	...

Query output (continued):

KEYSTORE_HOST	KEYSTORE_IP	KEYSTORE_IP_TYPE
hotel70.torolab.ibm.com	9.26.121.28	IPV4

- Retrieve information about the last master key rotation operation.

```
SELECT PREVIOUS_MASTER_KEY_LABEL, AUTH_ID, APPL_ID, ROTATION_TIME
FROM TABLE(SYSPROC.ADMIN_GET_ENCRYPTION_INFO())
```

This query returns the following output:

PREVIOUS_MASTER_KEY_LABEL	AUTH_ID	...
DB2_SYSGEN_geoffrey_SAMPLE_2013-12-10-02.12.34	GEOFFREY	...

Query output (continued):

APPL_ID	ROTATION_TIME
*LOCAL.geoffrey.140104183255	2013-12-11-09.23.56.000000

- Retrieve information about the PKCS #11 keystore:

```
SELECT KEYSTORE_HOST FROM TABLE(SYSPROC.ADMIN_GET_ENCRYPTION_INFO())
```

This query returns the following output:

```
DB2partition ;Safenet, Inc. ;LunaSA ;1146376383692 ;
```

## ADMIN\_GET\_INDEX\_COMPRESS\_INFO table function - returns compressed index information

The ADMIN\_GET\_INDEX\_COMPRESS\_INFO table function returns the potential index compression savings for uncompressed indexes.

### Syntax

```
►►—ADMIN_GET_INDEX_COMPRESS_INFO—(—objecttype—,—objectschema—,—objectname—,—
►—member—,—datapartitionid—)—————►
```

The schema is SYSPROC.

### Table function parameters

#### objecttype

An input argument of type VARCHAR(1) that indicates the object type. The value must be one of the following case-sensitive values:

- 'T', NULL, or the empty string to indicate a table
- 'I' for an index

#### objectschema

A case-sensitive input parameter of type VARCHAR(128) that specifies the object schema.

If *objecttype* is 'T', NULL, or the empty string (''), then *objectschema* indicates the table schema.

- If *objectschema* is specified and *objectname* is NULL or the empty string (''), then information is returned for all indexes on all tables in the specified schema.
- If both *objectschema* and *objectname* are specified, then information is returned for all indexes on the specified table.

If *objecttype* is 'I', then *objectschema* indicates the index schema.

- If *objectschema* is specified and *objectname* is NULL or the empty string ("), then information is returned for all indexes in the specified schema.
- If both *objectschema* and *objectname* are specified, then information is returned for the specified index.
- If neither *objectschema* or *objectname* are specified, then information is returned for all indexes in all of the schemas.

If *objectname* is specified and *objectschema* is not specified, the function returns an SQL error. A parameter value is said to be unspecified when either it has a value of NULL or the empty string (").

#### *objectname*

A case-sensitive input parameter of type VARCHAR(128) that specifies the object name. See the description for the *objectschema* parameter.

#### *member*

An input parameter of type INTEGER that specifies a database member number. When specified, information is returned only for indexes that reside on the specified database member. To specify that data should be returned for all active database members, set the *member* parameter value to either -2 or NULL. In single-member environments, specify -2 or NULL.

#### *datapartitionid*

An input parameter of type INTEGER that specifies the data partition ID. When specified, information is returned only for index partitions defined on the specified data partitions. The data partition ID should correspond to the DATAPARTITIONID found in the SYSCAT.DATAPARTITIONS view. To specify that data should be returned for all data partitions, set the *datapartitionid* parameter value to either -2 or NULL. For nonpartitioned indexes, specify -2, 0, or NULL.

## Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Usage notes

- Do not use this function on existing indexes to get actual compression savings; instead run the RUNSTATS command (if statistics are not up to date), then query either or both of the following catalog values:
  - SYSCAT.INDEXES.PCTPAGESSAVED
  - SYSCAT.INDEXPARTITIONS.PCTPAGESSAVED

## Example

After database migration, all the existing indexes are uncompressed. You may want to estimate the potential index compression savings for existing indexes on the

table "S.T1", which has a data partition ID of 3 and resides on database partition number 2. In this example, S is the schema name and T1 is the table name, and T1 is not compressed

```
SELECT compress_attr, iid, dbpartitionnum, index_compressed,
 pct_pages_saved, num_leaf_pages_saved
FROM TABLE(sysproc.admin_get_index_compress_info('', 'S', 'T1', 2, 3))
AS t
```

The following is a sample of the output from this statement.

COMPRESS_ATTR	IID	DBPARTITIONNUM	INDEX_COMPRESSED	...
N	1	2	N	...
N	2	2	N	...
...	PCT_PAGES_SAVED	NUM_LEAF_PAGES_SAVED		
...	50	200		
...	45	150		

You may decide that the savings from compression are worthwhile, and you want to enable index compression.

```
ALTER INDEX INDEX1 compress yes
ALTER INDEX INDEX2 compress yes
REORG INDEXES all FOR table S.T1
```

As time passes, you may determine the need to create new indexes for the table and want to estimate index compression savings for these indexes before compressing them. You may also want to see the compression statistics from already compressed indexes.

```
SELECT compress_attr, iid, dbpartitionnum, index_compressed,
 pct_pages_saved, num_leaf_pages_saved
FROM TABLE(sysproc.admin_get_index_compress_info('', 'S', 'T1', 2, 3))
AS t
```

The following is a sample of the output from this statement.

COMPRESS_ATTR	IID	DBPARTITIONNUM	INDEX_COMPRESSED	...
Y	1	2	Y	...
Y	2	2	Y	...
N	3	2	N	...
N	4	2	N	...
...	PCT_PAGES_SAVED	NUM_LEAF_PAGES_SAVED		
...	-1	-1		
...	-1	-1		
...	58	230		
...	49	140		

As the first two indexes were already compressed, as indicated by the index\_compressed column, the statement returns values from the system catalogs. In this case, the values from the catalogs were not collected.

After running RUNSTATS on the table, the next run of the index function yields the corrected results.

```
RUNSTATS ON TABLE S.T1 FOR INDEXES ALL
SELECT compress_attr, iid, dbpartitionnum, index_compressed,
 pct_pages_saved, num_leaf_pages_saved
FROM TABLE(sysproc.admin_get_index_compress_info('', 'S', 'T1', 2, 3))
AS t
```

The following is a sample of the output from this statement.

```

COMPRESS_ATTR IID DBPARTITIONNUM INDEX_COMPRESSED ...

Y 1 2 Y ...
Y 2 2 Y ...
N 3 2 N ...
N 4 2 N ...
... PCT_PAGES_SAVED NUM_LEAF_PAGES_SAVED
... -----
... 50 200
... 45 150
... 58 230
... 49 140

```

### Information returned

Table 33. Information returned by ADMIN\_GET\_INDEX\_COMPRESS\_INFO

Column Name	Data Type	Description
INDSCHEMA	VARCHAR(128)	index_schema - Index schema monitor element
INDNAME	VARCHAR(128)	index_name - Index name monitor element
TABSCHEMA	VARCHAR(128)	table_schema - Table schema name monitor element
TABNAME	VARCHAR(128)	table_name - Table name monitor element
DBPARTITIONNUM	SMALLINT	dbpartitionnum - Database partition number monitor element
IID	SMALLINT	iid - Index identifier monitor element
DATAPARTITIONID	INTEGER	Data partition ID.
COMPRESS_ATTR	CHAR(1)	The state of the COMPRESSION attribute on the index. <ul style="list-style-type: none"> <li>“Y” = Index compression is enabled</li> <li>“N” = Index compression is not enabled</li> </ul>
INDEX_COMPRESSED	CHAR(1)	Physical index format. <ul style="list-style-type: none"> <li>“Y” = Index is in compressed format</li> <li>“N” = Index is in uncompressed format</li> </ul> If the physical index format does not match the compression attribute, an index reorganization is needed to convert index to the defined format. If the table or index is in error at the time this function is executed, then this value is NULL.
PCT_PAGES_SAVED	SMALLINT	If the index is not physically compressed (INDEX_COMPRESSED is “N”), then this value represents the estimated percentage of leaf pages saved, as if the index were actually compressed. If the index is physically compressed (INDEX_COMPRESSED is “Y”), then this value reports the PCTPAGESSAVED value from the system catalog view (either SYSCAT.INDEXES or SYSCAT.INDEXPARTITIONS). <p><b>Note:</b> This value is the same for each entry of an index or index partition for each database partition in a partitioned database environment. If the table or index is in error at the time this function is executed, then this value is NULL.</p>

Table 33. Information returned by ADMIN\_GET\_INDEX\_COMPRESS\_INFO (continued)

Column Name	Data Type	Description
NUM_LEAF_PAGES_SAVED	BIGINT	<p>If the index is not physically compressed (INDEX_COMPRESSED is "N"), then this value represents the estimated number of leaf pages saved as if the index were actually compressed. If the index is physically compressed (INDEX_COMPRESSED is "Y"), then this value reports the calculated number of leaf pages saved, based on the PCTPAGESSAVED and NLEAF values from the system catalog view (either SYSCAT.INDEXES or SYSCAT.INDEXPARTITIONS). If either PCTPAGESSAVED or NLEAF are invalid values (-1), then this value is set to -1 as well.</p> <p><b>Note:</b> This value is the same for each entry of an index or index partition for each database partition in a partitioned database environment. If the table or index is in error at the time this function is executed, then this value is NULL.</p>

## ADMIN\_GET\_INDEX\_INFO table function - returns index information

The ADMIN\_GET\_INDEX\_INFO table function returns index information not available in the catalog views, such as compression information and the logical and physical size of the index.

### Syntax

►►—ADMIN\_GET\_INDEX\_INFO—(—*objecttype*—,—*objectschema*—,—*objectname*—)—►►

The schema is SYSPROC.

### Table function parameters

#### *objecttype*

An input argument of type VARCHAR(1) that indicates the object type. The value must be one of the following case-sensitive values:

- 'T', NULL, or the empty string (") to indicate a table
- 'I' for an index

#### *objectschema*

A case-sensitive input parameter of type VARCHAR(128) that specifies the object schema.

If *objecttype* is 'T', NULL, or the empty string ("), then *objectschema* indicates the table schema.

- If *objectschema* is specified and *objectname* is NULL or the empty string ("), then information is returned for all indexes on all tables in the specified schema.
- If both *objectschema* and *objectname* are specified, then information is returned for all indexes on the specified table.

If *objecttype* is 'I', then *objectschema* indicates the index schema.

- If *objectschema* is specified and *objectname* is NULL or the empty string ("), then information is returned for all indexes in the specified schema.
- If both *objectschema* and *objectname* are specified, then information is returned for the specified index.

- If neither *objectschema* or *objectname* are specified, then information is returned for all indexes in all of the schemas.

If *objectname* is specified and *objectschema* is not specified, the function returns an SQL error. A parameter value is said to be unspecified when either it has a value of NULL or the empty string (").

#### *objectname*

A case-sensitive input parameter of type VARCHAR(128) that specifies the object name. See the description for the *objectschema* parameter.

### Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

### Example

After enabling index compression for several indexes on a table, you want to determine which indexes are compressed and which indexes require a rebuild in order to be compressed. In this example, S is the schema name and T1 is the table name.

```
db2 SELECT iid, compress_attr, index_compressed
 FROM TABLE(sysproc.admin_get_index_info('', 'S', 'T1')) AS t
```

The following is an example of output from this query.

IID	COMPRESS_ATTR	INDEX_COMPRESSED
1	Y	Y
2	Y	Y
3	Y	N
4	N	N

Additionally, you want to see other index information for all indexes in the schema S2. In this example:

- T2 = a partitioned table with two data partitions
- T3 = a nonpartitioned table
- IND\_1 = a nonpartitioned index on T2
- IND\_2 = a partitioned index on T2
- IND\_3 = a partitioned index on T2
- IND\_4 = an index on T3
- IND\_5 = an index on T3

```
db2 SELECT tabname, indname, iid, index_partitioning, datapartitionid,
 index_object_l_size, index_object_p_size, index_requires_rebuild,
 large_rids FROM TABLE(sysproc.admin_get_index_info('I', 'S2', '')) AS t
```

The following is an example of the output from this query.

TABNAME	INDNAME	IID	INDEX_PARTITIONING	DATAPARTITIONID
T2	IND_1	1	N	0
T2	IND_2	2	P	1
T2	IND_2	2	P	2
T2	IND_3	3	P	1
T2	IND_3	3	P	2
T3	IND_4	4		0
T3	IND_5	5		0

Output from this procedure (continued):

INDEX_OBJECT_L_SIZE	INDEX_OBJECT_P_SIZE	INDEX_REQUIRES_REBUILD	LARGE_RIDS
50	51	N	Y
40	40	N	Y
45	45	N	Y
40	40	N	Y
45	45	N	Y
20	20	N	Y
20	20	N	Y

### Information returned

Table 34. Information returned by ADMIN\_GET\_INDEX\_INFO

Column Name	Data Type	Description
INDSCHEMA	VARCHAR(128)	index_schema - Index schema monitor element
INDNAME	VARCHAR(128)	index_name - Index name monitor element
TABSCHEMA	VARCHAR(128)	table_schema - Table schema name monitor element
TABNAME	VARCHAR(128)	table_name - Table name monitor element
DBPARTITIONNUM	SMALLINT	dbpartitionnum - Database partition number monitor element
IID	SMALLINT	iid - Index identifier monitor element
DATAPARTITIONID	INTEGER	Data partition ID.
COMPRESS_ATTR	CHAR(1)	The state of the COMPRESSION attribute on the index. <ul style="list-style-type: none"> <li>“Y” = Index compression is enabled</li> <li>“N” = Index compression is not enabled</li> </ul>
INDEX_COMPRESSED	CHAR(1)	Physical index format. <ul style="list-style-type: none"> <li>“Y” = Index is in compressed format</li> <li>“N” = Index is in uncompressed format</li> </ul> If the physical index format does not match the compression attribute, an index reorganization is needed to convert the index to the defined format. If the table or index is in error when this function is executed, then this value is NULL.
INDEX_PARTITIONING	CHAR(1)	Identifies the partitioning characteristic of the index. <ul style="list-style-type: none"> <li>“N” = Nonpartitioned index</li> <li>“P” = Partitioned index</li> <li>Blank = Index is not on a partitioned table</li> </ul>

Table 34. Information returned by ADMIN\_GET\_INDEX\_INFO (continued)

Column Name	Data Type	Description
INDEX_OBJECT_L_SIZE	BIGINT	<p>Logical size of the index object. For nonpartitioned tables, this is the amount of disk space logically allocated for all indexes defined on the table. For a nonpartitioned index on a partitioned table, this is the amount of disk space logically allocated for the index. For a partitioned index on a partitioned table, this is the amount of disk space logically allocated for all index partitions defined on the data partition. All sizes are reported in kilobytes (KB).</p> <p>The logical size is the amount of space that the table or data partition knows about. It may be less than the amount of space physically allocated to hold index data for the table or data partition (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for the indexes and, for indexes created in DMS table spaces, an estimate of the EMP extents. If the table or index is in error when this function is executed, then this value is NULL.</p>
INDEX_OBJECT_P_SIZE	BIGINT	<p>Physical size of the index object. For nonpartitioned tables, this is the amount of disk space physically allocated for all indexes defined on the table. For a nonpartitioned index on a partitioned table, this is the amount of disk space physically allocated for the index. For a partitioned index on a partitioned table, this is the amount of disk space physically allocated for all index partitions defined on the data partition. All sizes are reported in kilobytes (KB).</p> <p>The size returned takes into account full extents allocated for the indexes and includes the EMP extents for indexes created in DMS table spaces. If the table or index is in error when this function is executed, then this value is NULL.</p>
INDEX_REQUIRES_REBUILD	CHAR(1)	<p>Rebuild status for the index.</p> <ul style="list-style-type: none"> <li>• “Y” if the index defined on the table or data partition requires a rebuild</li> <li>• “N” otherwise</li> </ul> <p>If the table is in error when this function is executed, then this value is NULL.</p>
LARGE_RIDS	CHAR(1)	<p>Indicates whether or not the index is using large row IDs (RIDs) (4 byte page number, 2 byte slot number).</p> <ul style="list-style-type: none"> <li>• “Y” indicates that the index is using large RIDs</li> <li>• “N” indicates that the index is not using large RIDs</li> <li>• “P” (pending) indicates that the table that the index is defined on supports large RIDs (that is, the table is in a large table space), but the index for the table or data partition has not been reorganized or rebuilt yet. Therefore, the table is still using 4 byte RIDs, and action must be taken to convert the table or index to large RIDs.</li> </ul> <p>If the table is in error where this function is executed, then this value is NULL.</p>

Table 34. Information returned by ADMIN\_GET\_INDEX\_INFO (continued)

Column Name	Data Type	Description
RECLAIMABLE_SPACE	BIGINT	This value applies only to an index in a DMS table space. This value is an estimate of disk space, in kilobytes, that can be reclaimed from the entire index object by running the <b>REORG INDEXES</b> or <b>REORG INDEX</b> command with the RECLAIM EXTENTS option. For any index not defined in a DMS table space, the value is zero. If the table or index is in error when this function is executed, then this value is NULL.

## ADMIN\_GET\_INTRA\_PARALLEL - Get intrapartition parallelism state

The ADMIN\_GET\_INTRA\_PARALLEL scalar function returns the current state of intrapartition parallelism for the application.

This function can be used to check if current statements are running with parallelized query access plan.

### Syntax

►►ADMIN\_GET\_INTRA\_PARALLEL—(—)—————►►

The schema is SYSPROC.

### Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Information returned

This function returns a value of type VARCHAR(3) that specifies the current state of intrapartition parallelism for the database application. One of the following values is returned:

- YES**     The database application will run with intrapartition parallelism enabled.
- NO**     The database application will run with intrapartition parallelism disabled.

### Usage notes

The value returned by ADMIN\_GET\_INTRA\_PARALLEL can be different from the state set by ADMIN\_SET\_INTRA\_PARALLEL in the following cases:

- ADMIN\_SET\_INTRA\_PARALLEL was called during a transaction and the transaction is not yet committed or rolled back.

- ADMIN\_SET\_INTRA\_PARALLEL was called during a transaction that opens a with hold cursor and the current transaction has not yet closed the cursor.
- The application is associated with a database workload, which has a value applied to the MAX DEGREE workload attribute that is different than the one specified by the call to ADMIN\_SET\_INTRA\_PARALLEL.

### Example

Find the current state of intrapartition parallelism from CLP:

```
VALUES SYSPROC.ADMIN_GET_INTRA_PARALLEL
```

```
1

NO
```

1 record(s) selected.

### ADMIN\_GET\_MEM\_USAGE table function - Get total memory consumption for instance

The ADMIN\_GET\_MEM\_USAGE table function gets the total memory consumption for a given instance.

### Syntax

```
➡➡ ADMIN_GET_MEM_USAGE ((member)) ➡➡
```

The schema is SYSPROC.

### Table function parameters

*member*

An optional input argument of type integer that specifies the member from which the memory usage statistics are retrieved. If -1 or the NULL value is specified, data is returned from the currently connected member.

### Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Information returned

Table 35. Information returned for ADMIN\_GET\_MEM\_USAGE

Column Name	Data type	Description
MEMBER	SMALLINT	member - Database member monitor element

Table 35. Information returned for ADMIN\_GET\_MEM\_USAGE (continued)

Column Name	Data type	Description
MAX_MEMBER_MEM	BIGINT	The maximum amount of memory (in bytes) available for the member.
CURRENT_MEMBER_MEM	BIGINT	The amount of memory (in bytes) currently used by the member.
PEAK_MEMBER_MEM	BIGINT	The peak or high watermark of memory (in bytes) used by the member since the instance started.

## Examples

*Example 1:* Report memory usage for all members

```
SELECT MEMBER, MAX_MEMBER_MEM, CURRENT_MEMBER_MEM, PEAK_MEMBER_MEM
FROM TABLE(SYSPROC.ADMIN_GET_MEM_USAGE()) AS T
```

MEMBER	MAX_MEMBER_MEM	CURRENT_MEMBER_MEM	PEAK_MEMBER_MEM
0	7430103040	958169088	958300160
3	7430103040	951615488	951615488
1	7430103040	952664064	952664064
2	7430103040	951615488	951615488

4 record(s) selected.

## ADMIN\_GET\_MSGS table function - Retrieve messages generated by a data movement utility that is executed through the ADMIN\_CMD procedure

The ADMIN\_GET\_MSGS table function is used to retrieve messages generated by a single execution of a data movement utility command through the ADMIN\_CMD procedure.

The input parameter *operation\_id* identifies that operation.

## Syntax

►► ADMIN\_GET\_MSGS (—*operation\_id*—) ◀◀

The schema is SYSPROC.

## Table function parameter

*operation\_id*

An input argument of type VARCHAR(139) that specifies the operation ID of the message file(s) produced by a data movement utility that was executed through the ADMIN\_CMD procedure. The operation ID is generated by the ADMIN\_CMD procedure.

## Authorization

EXECUTE privilege on the ADMIN\_GET\_MSGS table function. The fenced user ID must have read access to the files under the directory indicated by registry variable **DB2\_UTIL\_MSGPATH**. If the registry variable is not set, then the fenced user ID must have read access to the files in the tmp subdirectory of the instance directory.

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Example

Check all the messages returned by EXPORT utility that was executed through ADMIN\_CMD procedure, with operation ID '24523\_THERESAX'

```
SELECT * FROM TABLE(SYSPROC.ADMIN_GET_MSGS('24523_THERESAX')) AS MSG
```

The following output is an example of sample output from this query.

DBPARTITIONNUM	AGENTTYPE	SQLCODE	MSG
-	-	SQL3104N	The Export utility is beginning to export data to file "/home/theresax/rtest/data/ac_load03.del".
-	-	SQL3105N	The Export utility has finished exporting "8" rows.

2 record(s) selected.

## Usage notes

The query statement that invokes this table function with the appropriate *operation\_id* can be found in the MSG\_RETRIEVAL column of the first result set returned by the ADMIN\_CMD procedure.

## Information returned

Table 36. Information returned by the ADMIN\_GET\_MSGS table function

Column name	Data type	Description
DBPARTITIONNUM	INTEGER	dbpartitionnum - Database partition number monitor element
AGENTTYPE	CHAR(4)	Agent type. This value is only returned for a distributed load. The possible values are: <ul style="list-style-type: none"><li>• 'LOAD': for load agent</li><li>• 'PART': for partitioning agent</li><li>• 'PREP': for pre-partitioning agent</li><li>• NULL: no agent type information is available</li></ul>
SQLCODE	VARCHAR(9)	SQLCODE of the message being returned.
MSG	VARCHAR(1024)	Short error message that corresponds to the SQLCODE.

## ADMIN\_GET\_STORAGE\_PATHS table function - retrieve automatic storage path information

The ADMIN\_GET\_STORAGE\_PATHS table function returns a list of automatic storage paths for each database storage group, including file system information for each storage path.

Refer to Table 37 on page 218 for a complete list of information that can be returned.

### Syntax

►►—ADMIN\_GET\_STORAGE\_PATHS—(—*storage\_group\_name*—,—*member*—)—►►

The schema is SYSPROC.

### Table function parameters

#### *storage\_group\_name*

An input argument of type VARCHAR(128) that specifies a valid storage group name in the currently connected database when this function is called. If the argument is NULL or an empty string, information is returned for all storage groups in the database. If the argument is specified, information is only returned for the identified storage group.

#### *member*

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function.

- In a Db2 pureScale, specify 0.
- In a partitioned database environment, specify -1 for the current database member, or -2 for all database members.

If the NULL value is specified, -1 is set implicitly.

### Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Example

Determine which dropped storage paths are still being used:

```
SELECT VARCHAR(STORAGE_GROUP_NAME, 30) AS STOGROUP, VARCHAR(DB_STORAGE_PATH, 40)
AS STORAGE_PATH FROM TABLE(ADMIN_GET_STORAGE_PATHS('',-1)) AS T
WHERE DB_STORAGE_PATH_STATE = 'DROP_PENDING'
```

The following is an example of the output from this query.

```

STOGROUP STORAGE_PATH

HOTSTORAGE /home/hotel55/hotpath1

1 record(s) selected.

```

List all the storage paths for the currently connected database:

```

SELECT VARCHAR(STORAGE_GROUP_NAME, 30) AS STOGROUP, VARCHAR(DB_STORAGE_PATH, 40)
 AS STORAGE_PATH FROM TABLE(ADMIN_GET_STORAGE_PATHS('','-1')) AS T

```

The following is an example of the output from this query.

```

STOGROUP STORAGE_PATH

IBMSTOGROUP /home/hotel55/instowner
HOTSTORAGE /home/hotel55/hotpath1
COLDSTORAGE /home/hotel55/coldpath1

3 record(s) selected.

```

## Information returned by ADMIN\_GET\_STORAGE\_PATHS

Table 37. Information returned by the ADMIN\_GET\_STORAGE\_PATHS table function

Column Name	Data Type	Description or corresponding monitor element
STORAGE_GROUP_NAME	VARCHAR(128)	storage_group_name - Storage group name
STORAGE_GROUP_ID	INTEGER	storage_group_id - Storage group identifier
DBPARTITIONNUM	SMALLINT	dbpartitionnum - Database partition number
DB_STORAGE_PATH	VARCHAR(256)	db_storage_path - Automatic storage path
DB_STORAGE_PATH_WITH_DPE	VARCHAR(256)	db_storage_path_with_dpe - Database storage path with database partition expression
DB_STORAGE_PATH_STATE	VARCHAR(16)	db_storage_path_state - Storage path state  Value is one of: <ul style="list-style-type: none"> <li>• IN_USE</li> <li>• NOT_IN_USE</li> <li>• DROP_PENDING</li> </ul>
DB_STORAGE_PATH_ID	BIGINT	db_storage_path_id - Storage path identifier
FS_ID	VARCHAR(22)	fs_id - Unique file system identification number
FS_TOTAL_SIZE	BIGINT	fs_total_size - Total size of a file system
FS_USED_SIZE	BIGINT	fs_used_size - Amount of space used on a file system
STO_PATH_FREE_SIZE	BIGINT	sto_path_free_size - Automatic Storage path free space

## ADMIN\_GET\_TAB\_COMPRESS\_INFO table function - estimate compression savings

The ADMIN\_GET\_TAB\_COMPRESS\_INFO table function estimates the compression savings that can be gained for the table, assuming a REORG with RESETDICTIONARY option will be performed.

This table function provides a direct replacement for the 'ESTIMATE' mode provided by the deprecated ADMIN\_GET\_TAB\_COMPRESS\_INFO table function in previous versions of Db2.

### Syntax

►►—ADMIN\_GET\_TAB\_COMPRESS\_INFO—(—*tabschema*—,—*tablename*—)————►►

The schema is SYSPROC.

### Table function parameters

#### *tabschema*

An input argument of type VARCHAR(128) that specifies the schema name.

#### *tablename*

An input argument of type VARCHAR(128) that specifies the table name, a materialized query table name or a hierarchy table name.

### Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Information returned

Table 38. Information returned for ADMIN\_GET\_TAB\_COMPRESS\_INFO

Column Name	Data Type	Description
TABSCHEMA	VARCHAR(128)	table_schema - Table schema name monitor element
TABNAME	VARCHAR(128)	table_name - Table name monitor element
DBPARTITIONNUM	SMALLINT	dbpartitionnum - Database partition number monitor element
DATAPARTITIONID	INTEGER	Data partition number.
OBJECT_TYPE	VARCHAR(4)	Type of object. One of: <ul style="list-style-type: none"><li>• DATA</li><li>• XML</li></ul>

Table 38. Information returned for ADMIN\_GET\_TAB\_COMPRESS\_INFO (continued)

Column Name	Data Type	Description
ROWCOMPMODE	CHAR(1)	The current row compression mode for the object. The returned metric can be one of the following values: <ul style="list-style-type: none"> <li>• 'S' if Classic Row Compression is enabled</li> <li>• 'A' if Adaptive Row Compression is enabled</li> <li>• Blank if no row compression is enabled</li> </ul>
PCTPAGESSAVED_CURRENT	SMALLINT	Current percentage of pages saved from row compression.
AVGROWSIZE_CURRENT	SMALLINT	Current average record length.
PCTPAGESSAVED_STATIC	SMALLINT	Estimated percentage of pages saved from Classic Row Compression.
AVGROWSIZE_STATIC	SMALLINT	Estimated average record length from Classic Row Compression.
PCTPAGESSAVED_ADAPTIVE	SMALLINT	Estimated percentage of pages saved from Adaptive Row Compression.
AVGROWSIZE_ADAPTIVE	SMALLINT	Estimated average record length from Adaptive Row Compression.

### Usage notes

- The ADMIN\_GET\_TAB\_COMPRESS\_INFO table function is specific to row compression and is not applicable to column-organized tables. The table function returns 0 rows if you specify a column-organized table.
- If both the *tabschema* and *tabname* are specified, information is returned for that specific table only.
- If the *tabschema* is specified but *tabname* is empty (") or NULL, information is returned for all tables in the given schema.
- If the *tabschema* is empty (") or NULL and *tabname* is specified, an error is returned. To retrieve information for a specific table, the table must be identified by both schema and table name.
- If both *tabschema* and *tabname* are empty (") or NULL, information is returned for all tables.
- If *tabschema* or *tabname* do not exist, or *tabname* does not correspond to a table name (type T) or a materialized query table name (type S), an empty result set is returned.
- When the ADMIN\_GET\_TAB\_COMPRESS\_INFO table function is retrieving data for a given table, it will acquire a shared lock on the corresponding row of SYSTABLES to ensure consistency of the data that is returned (for example, to ensure that the table is not altered while information is being retrieved for it). The lock will only be held for as long as it takes to retrieve the compression information for the table, and not for the duration of the table function call.
- If the specified table has one or more XML columns, the ADMIN\_GET\_TAB\_COMPRESS\_INFO table function will return two rows per partition. One row with OBJECT\_TYPE returning 'DATA' and another row with OBJECT\_TYPE returning 'XML'. If the specified table does not have any XML columns, then only one row per partition will be returned with OBJECT\_TYPE of 'DATA'.

- For XML object types, the estimates returned for PCTPAGESSAVED\_ADAPTIVE and PCTPAGESSAVED\_STATIC are identical as adaptive compression only applies to the data portion of the table.

### Examples

*Example 1:* View the current compression results and estimate report of both classic row compression and adaptive compression information of the TABLE1 table in the SCHEMA1 schema.

```
SELECT SUBSTR(TABSCHEMA, 1, 10) AS TABSCHEMA, SUBSTR(TABNAME, 1, 10) AS TABNAME,
 DBPARTITIONNUM, DATAPARTITIONID, OBJECT_TYPE, ROWCOMPMODE,
 PCTPAGESSAVED_CURRENT, AVGROWSIZE_CURRENT,
 PCTPAGESSAVED_STATIC, AVGROWSIZE_STATIC,
 PCTPAGESSAVED_ADAPTIVE, AVGROWSIZE_ADAPTIVE
FROM TABLE(SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO('SCHEMA1', 'TABLE1'))
```

Output from this query:

TABSCHEMA	TABNAME	DBPARTITIONNUM	DATAPARTITIONID	OBJECT_TYPE	ROWCOMPMODE	...
SCHEMA1	TABLE1	0	0	DATA	A	...
SCHEMA1	TABLE1	0	0	XML	S	...

PCTPAGESSAVED_CURRENT	AVGROWSIZE_CURRENT	PCTPAGESSAVED_STATIC	AVGROWSIZE_STATIC	...
	60	40	68	34 ...
	58	255	62	198 ...

PCTPAGESSAVED_ADAPTIVE	AVGROWSIZE_ADAPTIVE	
	70	30
	62	198

2 record(s) selected.

**ADMIN\_GET\_TAB\_DICTIONARY\_INFO**  
**table function - report properties of existing table dictionaries**  
 The ADMIN\_GET\_TAB\_DICTIONARY\_INFO table function reports the dictionary information of classic row compression for a specified schema and table when the table dictionary was created.

This is a direct replacement for the 'REPORT' mode provided by the deprecated ADMIN\_GET\_TAB\_COMPRESS\_INFO table function in previous versions of Db2.

### Syntax

```
►►—ADMIN_GET_TAB_DICTIONARY_INFO—(—tabschema—,—tablename—)—————►
```

The schema is SYSPROC.

### Table function parameters

- tabschema*  
 An input argument of type VARCHAR(128) that specifies the schema name.
- tablename*  
 An input argument of type VARCHAR(128) that specifies the table name, a materialized query table name or a hierarchy table name.

## Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

None

## Usage notes

- The ADMIN\_GET\_TAB\_DICTIONARY\_INFO table function is specific to row compression and is not applicable to column-organized tables. The table function returns 0 rows if you specify a column-organized table.
- If both the *tabschema* and *tabname* are specified, information is returned for that specific table only.
- If the *tabschema* is specified but *tabname* is empty (") or NULL, information is returned for all tables in the given schema.
- If the *tabschema* is empty (") or NULL and *tabname* is specified, an error is returned. To retrieve information for a specific table, the table must be identified by both schema and table name.
- If both *tabschema* and *tabname* are empty (") or NULL, information is returned for all tables.
- If *tabschema* or *tabname* do not exist, or *tabname* does not correspond to a table name (type T) or a materialized query table name (type S), an empty result set is returned.
- If the specified table has one or more XML columns, the ADMIN\_GET\_TAB\_DICTIONARY\_INFO table function will return two rows per partition. One row with OBJECT\_TYPE returning 'DATA' and another row with OBJECT\_TYPE returning 'XML'. If the specified table does not have any XML columns, then only one row per partition will be returned with OBJECT\_TYPE of 'DATA'.
- When the ADMIN\_GET\_TAB\_DICTIONARY\_INFO table function is retrieving data for a given table, it will acquire a shared lock on the corresponding row of SYSTABLES to ensure consistency of the data that is returned (for example, to ensure that the table is not altered while information is being retrieved for it). The lock will only be held for as long as it takes to retrieve the compression information for the table, and not for the duration of the table function call.

## Information returned

Table 39. Information returned by ADMIN\_GET\_TAB\_DICTIONARY\_INFO

Column Name	Data Type	Description
TABSCHEMA	VARCHAR(128)	table_schema - Table schema name monitor element
TABNAME	VARCHAR(128)	table_name - Table name monitor element
DBPARTITIONNUM	SMALLINT	dbpartitionnum - Database partition number monitor element
DATAPARTITIONID	INTEGER	Data partition number.

Table 39. Information returned by ADMIN\_GET\_TAB\_DICTIONARY\_INFO (continued)

Column Name	Data Type	Description
OBJECT_TYPE	VARCHAR(4)	Type of object. One of: <ul style="list-style-type: none"> <li>• DATA</li> <li>• XML</li> </ul>
ROWCOMPMODE	CHAR(1)	The current row compression mode for the object. The returned metric can be one of the following values: <ul style="list-style-type: none"> <li>• 'S' if Classic Row Compression is enabled</li> <li>• 'A' if Adaptive Row Compression is enabled</li> <li>• Blank if no row compression is enabled</li> </ul>
BUILDER	VARCHAR(30)	Code path taken to build the dictionary, which can be one of the following values: <ul style="list-style-type: none"> <li>• 'INSPECT' = INSPECT ROWCOMPESTIMATE</li> <li>• 'LOAD' = LOAD INSERT/REPLACE</li> <li>• 'NOT BUILT' = no dictionary available</li> <li>• 'REDISTRIBUTE' = REDISTRIBUTE</li> <li>• 'REORG' = REORG RESETDICTIONARY</li> <li>• 'TABLE GROWTH' = INSERT</li> </ul>
BUILD_TIMESTAMP	TIMESTAMP	Timestamp of when the dictionary was built. Timestamp granularity is to the second. If no dictionary is available, then the timestamp is NULL.
SIZE	BIGINT	Size of the expansion dictionary measured in bytes. If a historical dictionary exists, this value is the sum of the current and historical dictionary sizes.
HISTORICAL_DICTIONARY	CHAR(1)	Indicates the presence of a historical dictionary. The returned metric can be one of the following values: <ul style="list-style-type: none"> <li>• 'N' = No historical dictionary exists</li> <li>• 'Y' = Historical dictionary exists</li> </ul>
ROWS_SAMPLED	BIGINT	Number of records that contributed to building the dictionary.
PCTPAGESSAVED	SMALLINT	Percentage of pages saved from compression. This information is a projection, based on the records contributing to building the dictionary.
AVGCOMPRESSEDROWSIZE	SMALLINT	Average compressed record length of the records contributing to building the dictionary.

Examples

Example 1. View a report of the dictionary information of the ADMIN\_VIEW table in the PAGECOMP schema.

```
SELECT SUBSTR(TABSCHEMA, 1, 10) AS TABSCHEMA, SUBSTR(TABNAME, 1, 10) AS TABNAME,
 DBPARTITIONNUM, DATAPARTITIONID, OBJECT_TYPE, ROWCOMPMODE, BUILDER,
 BUILD_TIMESTAMP, SIZE, HISTORICAL_DICTIONARY, ROWS_SAMPLED,
 PCTPAGESSAVED, AVGCOMPRESSEDROWSIZE
FROM TABLE(SYSPROC.ADMIN_GET_TAB_DICTIONARY_INFO('PAGECOMP', 'ADMIN_VIEW'))
```

Output from this query:

TABSCHEMA	TABNAME	DBPARTITIONNUM	DATAPARTITIONID	OBJECT_TYPE	ROWCOMPMODE	...
PAGECOMP	ADMIN_VIEW	0	0	DATA	S	...
BUILDER		BUILD_TIMESTAMP		SIZE		
REORG		2010-09-03-01.10.33.000000		52736		
HISTORICAL_DICTIONARY		ROWS_SAMPLED	PCTPAGESSAVED	AVGCOMPRESSEDROWSIZE		
N		300000	80	38		

1 record(s) selected.

ADMINTABINFO administrative view and ADMIN\_GET\_TAB\_INFO table function - retrieve table size and state information

The ADMINTABINFO administrative view and the ADMIN\_GET\_TAB\_INFO table function provide methods to retrieve table size and state information that is not currently available in the catalog views.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “ADMINTABINFO administrative view”
- “ADMIN\_GET\_TAB\_INFO table function” on page 225

ADMINTABINFO administrative view

The ADMINTABINFO administrative view returns size and state information for tables, materialized query tables (MQT) and hierarchy tables only. These table types are reported as T for table, S for materialized query tables and H for hierarchy tables in the SYSCAT.TABLES catalog view. The information is returned at both the data partition level and the database partition level for a table.

In a Db2 pureScale environment, values reported for a table are identical on all members since all members operate on a single physical partition of data. This is unlike in a partitioned database environment, where each member operates on a different physical partition of data, and reports different values. Because the values on all members are always the same, the ADMINTABINFO view and the ADMIN\_GET\_TAB\_INFO table functions return only a single row for each table when run on a Db2 pureScale instance.

The schema is SYSIBMADM.

Refer to the Information returned by ADMINTABINFO administrative view and ADMIN\_GET\_TAB\_INFO table for a complete list of information that can be returned.

## Authorization

One of the following authorizations is required:

- SELECT privilege on the ADMINTABINFO administrative view
- CONTROL privilege on the ADMINTABINFO administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

## Examples

*Example 1:* Retrieve size and state information for all tables

```
SELECT * FROM SYSIBMADM.ADMINTABINFO
```

*Example 2:* Determine the amount of physical space used by a large number of sparsely populated tables.

```
SELECT TABSCHEMA, TABNAME, SUM(DATA_OBJECT_P_SIZE),
 SUM(INDEX_OBJECT_P_SIZE), SUM(LONG_OBJECT_P_SIZE),
 SUM(LOB_OBJECT_P_SIZE), SUM(XML_OBJECT_P_SIZE)
FROM SYSIBMADM.ADMINTABINFO GROUP BY TABSCHEMA, TABNAME
```

*Example 3:* Identify tables that are eligible to use large RIDs, but are not currently enabled to use large RIDs.

```
SELECT TABSCHEMA, TABNAME FROM SYSIBMADM.ADMINTABINFO
WHERE LARGE_RIDS = 'P'
```

*Example 4:* Check the current type of statistics information collected for table T1

```
SELECT SUBSTR(TABSCHEMA, 1, 10) AS TBSchema, SUBSTR(TABNAME, 1, 10)
 AS TBNAME, STATTYPE FROM SYSIBMADM.ADMINTABINFO WHERE TABNAME = 'T1';
```

TBSchema	TBNAME	STATTYPE
DB2USER1	T1	U

1 record(s) selected.

## ADMIN\_GET\_TAB\_INFO table function

The ADMIN\_GET\_TAB\_INFO table function returns the same information as the ADMINTABINFO administrative view, but allows you to specify a schema and table name.

Refer to the Information returned by ADMINTABINFO administrative view and ADMIN\_GET\_TAB\_INFO table for a complete list of information that can be returned.

## Authorization

EXECUTE privilege on the ADMIN\_GET\_TAB\_INFO table function.

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Syntax

►►—ADMIN\_GET\_TAB\_INFO—(—*tabschema*—,—*tabname*—)—►►

The schema is SYSPROC.

## Routine parameters

### *tabschema*

An input argument of type VARCHAR(128) that specifies a schema name.

### *tabname*

An input argument of type VARCHAR(128) that specifies a table name, a materialized query table name or a hierarchy table name.

## Information returned

Table 40. Information Returned by ADMIN\_TABINFO administrative view and ADMIN\_GET\_TAB\_INFO

Column name	Data type	Description
TABSCHEMA	VARCHAR(128)	table_schema - Table schema name monitor element
TABNAME	VARCHAR(128)	table_name - Table name monitor element
TABTYPE	CHAR(1)	Table type: <ul style="list-style-type: none"><li>• 'H' = hierarchy table</li><li>• 'S' = materialized query table</li><li>• 'T' = table</li></ul>
DBPARTITIONNUM	SMALLINT	dbpartitionnum - Database partition number monitor element
DATA_PARTITION_ID	INTEGER	data_partition_id - Data partition identifier monitor element
AVAILABLE	CHAR(1)	State of the table: <ul style="list-style-type: none"><li>• 'N' = the table is unavailable. If the table is unavailable, all other output columns relating to the size and state will be NULL.</li><li>• 'Y' = the table is available.</li></ul> <b>Note:</b> Rollforward through an unrecoverable load will put a table into the unavailable state.

Table 40. Information Returned by ADMINTABINFO administrative view and ADMIN\_GET\_TAB\_INFO (continued)

Column name	Data type	Description
DATA_OBJECT_L_SIZE	BIGINT	<p>Data object logical size. Amount of disk space logically allocated for the table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated for the table (for example, in the case of a logical table truncation). For multidimensional clustering (MDC) and insert time clustering (ITC) tables, this size includes the logical size of the block map object. The size returned takes into account full extents that are logically allocated for the table and, for objects created in DMS table spaces, an estimate of the Extent Map Page (EMP) extents. This size represents the logical size of the base table only. Space consumed by LOB data, Long Data, Indexes and XML objects are reported by other columns.</p>
DATA_OBJECT_P_SIZE	BIGINT	<p>Data object physical size, reported in kilobytes.</p> <p>For row-organized tables, the data object physical size is the amount of disk space that is physically allocated for the table. For MDC and ITC tables, this size includes the size of the block map object. The size that is returned takes into account full extents that are allocated for the table and includes the EMP extents for objects that you create in DMS table spaces. This size represents the physical size of the base table only. The space that is consumed by LOB data, long data, indexes, and XML objects is reported in other columns.</p> <p>For column-organized tables, the data object physical size is the amount of disk space that is physically allocated for the table metadata only, which is relatively small. The user data for column-organized tables is stored in the column-organized data object, and its physical size is reported by COL_OBJECT_P_SIZE.</p>
INDEX_OBJECT_L_SIZE	BIGINT	<p>Index object logical size. Amount of disk space logically allocated for the indexes defined on the table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold index data for the table (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for the indexes and, for indexes created in DMS table spaces, an estimate of the EMP extents.</p> <p>For partitioned indexes on partitioned tables, this is the logical size of the index object containing index partitions for the data partition identified by DATA_PARTITION_ID. This value does not take into account nonpartitioned indexes on partitioned tables. For information about the both partitioned and nonpartitioned indexes, you can use the ADMIN_GET_INDEX_INFO function.</p>

Table 40. Information Returned by ADMIN\_TABINFO administrative view and ADMIN\_GET\_TAB\_INFO (continued)

Column name	Data type	Description
INDEX_OBJECT_P_SIZE	BIGINT	<p>Index object physical size. Amount of disk space physically allocated for the indexes defined on the table, reported in kilobytes. The size returned takes into account full extents allocated for the indexes and includes the EMP extents for indexes created in DMS table spaces.</p> <p>For partitioned indexes on partitioned tables, this is the physical size of the index object containing index partitions for the data partition identified by DATA_PARTITION_ID. This value does not take into account nonpartitioned indexes on partitioned tables. For information about both partitioned and nonpartitioned indexes, you can use the ADMIN_GET_INDEX_INFO function.</p>
LONG_OBJECT_L_SIZE	BIGINT	<p>Long object logical size. Amount of disk space logically allocated for long field data in a table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold long field data for the table (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for long field data and, for long field data created in DMS table spaces, an estimate of the EMP extents.</p>
LONG_OBJECT_P_SIZE	BIGINT	<p>Long object physical size. Amount of disk space physically allocated for long field data in a table, reported in kilobytes. The size returned takes into account full extents allocated for long field data and includes the EMP extents for long field data created in DMS table spaces.</p>
LOB_OBJECT_L_SIZE	BIGINT	<p>LOB object logical size. Amount of disk space logically allocated for LOB data in a table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold LOB data for the table (for example, in the case of a logical table truncation). The size includes space logically allocated for the LOB allocation object. The size returned takes into account full extents that are logically allocated for LOB data and, for LOB data created in DMS table spaces, an estimate of the EMP extents.</p>
LOB_OBJECT_P_SIZE	BIGINT	<p>LOB object physical size. Amount of disk space physically allocated for LOB data in a table, reported in kilobytes. The size includes space allocated for the LOB allocation object. The size returned takes into account full extents allocated for LOB data and includes the EMP extents for LOB data created in DMS table spaces.</p>
XML_OBJECT_L_SIZE	BIGINT	<p>XML object logical size. Amount of disk space logically allocated for XML data in a table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold XML data for the table (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for XML data and, for XML data created in DMS table spaces, an estimate of the EMP extents.</p>

Table 40. Information Returned by ADMIN\_TABINFO administrative view and ADMIN\_GET\_TAB\_INFO (continued)

Column name	Data type	Description
XML_OBJECT_P_SIZE	BIGINT	XML object physical size. Amount of disk space physically allocated for XML data in a table, reported in kilobytes. The size returned takes into account full extents allocated for XML data and includes the EMP extents for XML data created in DMS table spaces.
INDEX_TYPE	SMALLINT	Indicates the type of indexes currently in use for the table. Returns 2 as type-2 indexes are used.
REORG_PENDING	CHAR(1)	A value of 'Y' indicates that a reorg recommended alter has been applied to the table and a classic (offline) reorg is required. Otherwise 'N' is returned.
INPLACE_REORG_STATUS	VARCHAR(10)	Current status of an inplace table reorganization on the table. The status value can be one of the following values: <ul style="list-style-type: none"> <li>ABORTED (in a PAUSED state, but unable to RESUME; STOP is required)</li> <li>EXECUTING</li> <li>NULL (if no inplace reorg has been performed on the table)</li> <li>PAUSED</li> </ul>
LOAD_STATUS	VARCHAR(12)	Current status of a load operation against the table. The status value can be one of the following values: <ul style="list-style-type: none"> <li>IN_PROGRESS</li> <li>NULL (if there is no load in progress for the table and the table is not in load pending state)</li> <li>PENDING</li> </ul>
READ_ACCESS_ONLY	CHAR(1)	'Y' if the table is in Read Access Only state, 'N' otherwise. A value of 'N' should not be interpreted as meaning that the table is fully accessible. If a load is in progress or pending, a value of 'Y' means the table data is available for read access, and a value of 'N' means the table is inaccessible. Similarly, if the table status is set integrity pending (refer to SYSCAT.TABLES STATUS column), then a value of 'N' means the table is inaccessible.
NO_LOAD_RESTART	CHAR(1)	A value of 'Y' indicates the table is in a partially loaded state that will not allow a load restart. A value of 'N' is returned otherwise.
NUM_REORG_REC_ALTERS	SMALLINT	Number of reorg recommend alter operations (for example, alter operations after which a reorganization is required) that have been performed against this table since the last reorganization.
INDEXES_REQUIRE_REBUILD	CHAR(1)	For nonpartitioned tables, 'Y' if any of the indexes defined on the table require a rebuild, and 'N' otherwise. For partitioned tables, 'Y' if any index partitions for the data partition identified by DATA_PARTITION_ID require a rebuild, and 'N' otherwise.

Table 40. Information Returned by ADMINTABINFO administrative view and ADMIN\_GET\_TAB\_INFO (continued)

Column name	Data type	Description
LARGE_RIDS	CHAR(1)	<p>Indicator of whether the table is using large row IDs (RID, a 4-byte page number and a 2-byte slot number). A value of 'Y' indicates that the table is using large RIDs, and 'N' indicates that it is not using large RIDs. A value of 'P' (pending) is returned if both the following criteria are met:</p> <ul style="list-style-type: none"> <li>• The table supports large RIDs, that is, the table is in a large table space.</li> <li>• At least one of the indexes for the table has not been reorganized or rebuilt yet, so the table is still using 4-byte RIDs. You must convert the table or indexes.</li> </ul> <p>This column returns the null value for column-organized tables because the concept of RIDs does not apply.</p>
LARGE_SLOTS	CHAR(1)	<p>Indicator of whether the table is using large slots, which allows more than 255 rows per page. A value of 'Y' indicates that the table is using large slots, and 'N' indicates that it is not using large slots. A value of 'P' (pending) is returned if the following criteria are both met:</p> <ul style="list-style-type: none"> <li>• The table supports large slots, that is, the table is in a large table space.</li> <li>• No offline table reorganization or table truncation operation has been performed on the table yet, so it is still using a maximum of 255 rows per page.</li> </ul> <p>This column returns the null value for column-organized tables because the concept of slots does not apply.</p>
DICTIONARY_SIZE	BIGINT	Size of the table dictionary, in bytes, used for row compression if a row compression dictionary exists for the table. If a historical dictionary exists, this value is the sum of the current and historical dictionary sizes.
BLOCKS_PENDING_CLEANUP	BIGINT	blocks_pending_cleanup - Pending cleanup rolled-out blocks monitor element

Table 40. Information Returned by ADMIN\_TABINFO administrative view and ADMIN\_GET\_TAB\_INFO (continued)

Column name	Data type	Description
STATSTYPE	CHAR(1)	<ul style="list-style-type: none"> <li>'F' = System fabricated statistics without table or index scan. These statistics are stored in memory and are different from what is stored in the system catalogs. This is a temporary state and eventually full statistics will be gathered by the database system and stored in the system catalogs.</li> <li>'A' = System asynchronously gathered statistics. Statistics have been automatically collected by the database system by a background process and stored in the system catalogs.</li> <li>'S' = System synchronously gathered statistics. Statistics have been automatically collected by the database system during SQL statement compilation. These statistics are stored in memory and are different from what is stored in the system catalogs. This is a temporary state and eventually the database system stores the statistics in the system catalogs.</li> <li>'U' = User gathered statistics. Statistics gathering was initiated by the user through a utility such as RUNSTATS, CREATE INDEX, LOAD, REDISTRIBUTE or by manually updating system catalog statistics.</li> <li>NULL = unknown type</li> </ul>
XML_RECORD_TYPE	SMALLINT	<p>Indicates the type of XML record currently in use for the table.</p> <ul style="list-style-type: none"> <li>1 if the type-1 (single node) XML record format is being used.</li> <li>2 if the type-2 (multi-node) XML record format is being used.</li> <li>Null if the table has no XML columns.</li> </ul>
RECLAIMABLE_SPACE	BIGINT	Indicates the amount of disk space that can be reclaimed by running the <b>REORG TABLE</b> command with the RECLAIM EXTENTS option for an MDC or ITC table in a DMS table space. For a column-organized table, this value indicates the amount of free space that was found the last time that the <b>RUNSTATS</b> command was run. Disk space is reported in kilobytes. For any other table, the value is zero.
XML_DICTIONARY_SIZE	BIGINT	Size of the XML dictionary, in bytes, used for data compression if a data compression dictionary exists for the XML storage object. If the table does not contain any XML columns or if a compression dictionary has not been created, the value is 0.
AMT_STATUS	VARCHAR(12)	Current status of ADMIN_MOVE_TABLE stored procedure call against the table. Returns the value 'IN_PROGRESS' or the null value if there is no move in progress for the table.
SPARSE_BLOCKS	BIGINT	For an insert time clustering (ITC) table, this field represents the number of sparsely used blocks in the table. This is the set of blocks that will be consolidated if the REORG TABLE command is invoked with the RECLAIM EXTENTS option.
STATS_ROWS_MODIFIED	BIGINT	stats_rows_modified - Rows modified since last RUNSTATS

Table 40. Information Returned by ADMIN\_TABINFO administrative view and ADMIN\_GET\_TAB\_INFO (continued)

Column name	Data type	Description
RTS_ROWS_MODIFIED	BIGINT	rts_rows_modified - Rows modified since last real time statistics
STATS_DBPARTITION	CHAR(3)	stats_dbpartition - Automatics statistics collection indicator
COL_OBJECT_L_SIZE	BIGINT	col_object_l_size - Column-organized data object logical size
COL_OBJECT_P_SIZE	BIGINT	col_object_p_size - Column-organized data object physical size

### Usage notes

- If both the *tabschema* and *tabname* are specified, information is returned for that specific table only.
- If the *tabschema* is specified but *tabname* is NULL or the empty string ("), then information is returned for all tables in the given schema.
- If the *tabschema* is NULL or the empty string (") and *tabname* is specified, then an error is returned. To retrieve information for a specific table, the table must be identified by both schema and table name.
- If both *tabschema* and *tabname* are NULL or the empty string ("), then information is returned for all tables.
- If *tabschema* or *tabname* do not exist, or *tabname* does not correspond to a table name (type T), a materialized query table name (type S) or a hierarchy table name (type H), an empty result set is returned.
- When the ADMIN\_GET\_TAB\_INFO table function is retrieving data for a given table, it will acquire a shared lock on the corresponding row of SYSTABLES to ensure consistency of the data that is returned (for example, to ensure that the table is not dropped while information is being retrieved for it). The lock will only be held for as long as it takes to retrieve the size and state information for the table, not for the duration of the table function call.
- Physical size reported for tables in SMS table spaces is the same as logical size.
- When an inplace reorg is active on a table, the physical size for the data object (DATA\_OBJECT\_P\_SIZE) will not be calculated. Only the logical size will be returned. You can tell if an inplace reorg is active on the table by looking at the INPLACE\_REORG\_STATUS output column.

### Examples

*Example 1:* Retrieve size and state information for the table DBUSER1.EMPLOYEE.

```
SELECT * FROM TABLE (SYSPROC.ADMIN_GET_TAB_INFO('DBUSER1', 'EMPLOYEE'))
AS T
```

*Example 2:* Suppose there exists a non-partitioned table (DBUSER1.EMPLOYEE), with all associated objects (for example, indexes and LOBs) stored in a single table space. Calculate how much physical space the table is using in the table space:

```
SELECT (data_object_p_size + index_object_p_size + long_object_p_size +
lob_object_p_size + xml_object_p_size + col_object_p_size) as total_p_size
FROM TABLE(SYSPROC.ADMIN_GET_TAB_INFO('DBUSER1', 'EMPLOYEE')) AS T
```

Calculate how much space would be required if the table were moved to another table space, where the new table space has the same page size and extent size as the original table space:

```
SELECT (data_object_l_size + index_object_l_size + long_object_l_size +
lob_object_l_size + xml_object_l_size + col_object_l_size) as total_l_size
FROM TABLE(SYSPROC.ADMIN_GET_TAB_INFO('DBUSER1', 'EMPLOYEE')) AS T
```

*Example 3:* Determine the total size for the compression dictionaries for the table DBUSER1.EMPLOYEE.

```
SELECT SUBSTR(TABSCHEMA,1,10) AS TBSCHHEMA, SUBSTR(TABNAME,1,10) AS TBNAME,
DICTIONARY_SIZE + XML_DICTIONARY_SIZE AS TOTAL_DICTIONARY_SIZE
FROM TABLE(SYSPROC.ADMIN_GET_TAB_INFO('DBUSER1','EMPLOYEE'))
```

*Example 4:* Determine the amount of space reclaimable from a multidimensional clustering table SAMPLE.STAFF:

```
SELECT RECLAIMABLE_SPACE
FROM TABLE(SYSPROC.ADMIN_GET_TAB_INFO('SAMPLE','STAFF'))
```

## **ADMINTEMPCOLUMNS administrative view and ADMIN\_GET\_TEMP\_COLUMNS table function - Retrieve column information for temporary tables**

The ADMINTEMPCOLUMNS administrative view and the ADMIN\_GET\_TEMP\_COLUMNS table function provide methods to retrieve column attribute information for created temporary tables and declared temporary tables.

Although the catalog views contain column attribute information for instances of created temporary tables, they do not have this information for declared temporary tables.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “ADMINTEMPCOLUMNS administrative view”
- “ADMIN\_GET\_TEMP\_COLUMNS table function” on page 234

### **ADMINTEMPCOLUMNS administrative view**

The ADMINTEMPCOLUMNS administrative view returns column attribute information for instances of created temporary tables and declared temporary tables.

The schema is SYSIBMADM.

Refer to the Information returned for ADMINTEMPCOLUMNS administrative view and ADMIN\_GET\_TEMP\_COLUMNS table for a complete list of information that can be returned.

### **Authorization**

One of the following authorizations is required:

- SELECT privilege on the ADMINTEMPCOLUMNS administrative view
- CONTROL privilege on the ADMINTEMPCOLUMNS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the ADMIN\_GET\_TEMP\_COLUMNS table function

- DATAACCESS authority

## Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

## Examples

*Example 1:* Retrieve column attribute information for all instances of created temporary tables and all declared temporary tables currently existing in the database.

```
SELECT * FROM SYSIBMADM.ADMINTEMPCOLUMNS
```

*Example 2:* Determine which temporary tables active in the database are using the user-defined data type of USMONEY.

```
SELECT APPLICATION_HANDLE, TABSCHEMA, TABNAME
FROM SYSIBMADM.ADMINTEMPCOLUMNS
WHERE TYPENAME = 'USMONEY'
```

*Example 3:* Retrieve table schema, table name, and the column names for all declared temporary tables declared by the SYSTEM\_USER.

```
SELECT T.TABSCHEMA, T.TABNAME, C.COLNAME
FROM SYSIBMADM.ADMINTEMPCOLUMNS C, SYSIBMADM.ADMINTEMPTABLES T
WHERE T.EMPTABTYPE = 'D'
AND T.INSTANTIATOR = SYSTEM_USER
AND T.TABSCHEMA = C.TABSCHEMA
AND T.TABNAME = C.TABNAME
```

## ADMIN\_GET\_TEMP\_COLUMNS table function

The ADMIN\_GET\_TEMP\_TABLES table function returns the same information as the ADMINTEMPCOLUMNS administrative view, but allows you to specify a schema name and a table name.

Refer to the Information returned for ADMINTEMPCOLUMNS administrative view and ADMIN\_GET\_TEMP\_COLUMNS table for a complete list of information that can be returned.

## Syntax

```
►►—ADMIN_GET_TEMP_COLUMNS—(—application_handle—,—tabschema—,—tabname—)—►◄
```

The schema is SYSPROC.

## Table function parameters

### *application\_handle*

An input argument of type BIGINT that specifies an application handle. If *application\_handle* is specified, data is returned for the specified connection only; if *application\_handle* is NULL, data is returned for all connections.

### *tabschema*

An input argument of type VARCHAR(128) that specifies a schema name.

*tabname*

An input argument of type VARCHAR(128) that specifies a created temporary table name or a declared temporary table name.

## Authorization

One of the following authorizations is required:

- EXECUTE privilege on the ADMIN\_GET\_TEMP\_COLUMNS table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Examples

*Example 1:* Retrieve column information for the declared temporary table TEMPEMPOLOYEE for the current connection.

```
SELECT *
 FROM TABLE (
 SYSPROC.ADMIN_GET_TEMP_COLUMNS(
 APPLICATION_ID(), 'SESSION', 'TEMPEMPOLOYEE'))
AS T
```

## Usage notes

- If both *tabschema* and *tabname* are specified, then information is returned for that specific temporary table only.
- If *tabschema* is specified but *tabname* is NULL or the empty string ("), then information is returned for all tables in the given schema.
- If *tabschema* is NULL or the empty string (") and *tabname* is specified, then an error is returned. To retrieve information for a specific temporary table, the table must be identified by both schema and table name.
- If both *tabschema* and *tabname* are NULL or the empty string ("), then information is returned for all temporary tables for the connection or all connections, depending on the value of *application\_handle*.
- If *tabschema* or *tabname* do not exist, or *tabname* does not correspond to a temporary table name, or instances of the identified temporary table do not exist in the database, then an empty result set is returned.

## Information returned

Table 41. Information returned for ADMINTEMPCOLUMNS administrative view and ADMIN\_GET\_TEMP\_COLUMNS

Column name	Data type	Description
APPLICATION_HANDLE	BIGINT	application_handle - Application handle monitor element
APPLICATION_NAME	VARCHAR(256)	appl_name - Application name monitor element
TABSCHEMA	VARCHAR(128)	table_schema - Table schema name monitor element
TABNAME	VARCHAR(128)	table_name - Table name monitor element
COLNAME	VARCHAR(128)	Name of the column.
COLNO	SMALLINT	Number of this column in the table (starting with 0).

Table 41. Information returned for ADMIN\_TEMP\_COLUMNS administrative view and ADMIN\_GET\_TEMP\_COLUMNS (continued)

Column name	Data type	Description
TYPE_SCHEMA	VARCHAR(128)	Schema name of the data type for the column.
TYPE_NAME	VARCHAR(128)	Unqualified name of the data type for the column.
LENGTH	INTEGER	Maximum length of the data. 0 for distinct types. The LENGTH column indicates precision for DECIMAL fields, and indicates the number of bytes of storage required for decimal floating-point columns; that is, 8 columns for DECFLOAT(16) and 16 columns for DECFLOAT(34).
SCALE	SMALLINT	Scale if the column type is DECIMAL; or the number of digits of fractional seconds if the column type is TIMESTAMP; 0 otherwise.
DEFAULT	VARCHAR(254)	Default value for the column of a table expressed as a constant, special register, or cast-function appropriate for the data type of the column. Can also be the keyword NULL. Values might be converted from what was specified as a default value. For example, date and time constants are shown in ISO format, cast-function names are qualified with schema names, and identifiers are delimited. Null value if a DEFAULT clause was not specified or the column is a view column.
NULLS	CHAR(1)	Nullability attribute for the column. <ul style="list-style-type: none"> <li>• "Y" = Column is nullable</li> <li>• "N" = Column is not nullable</li> </ul> <p>The value can be "N" for a view column that is derived from an expression or function. Nevertheless, such a column allows null values when the statement using the view is processed with warnings for arithmetic errors.</p>
CODEPAGE	SMALLINT	Code page used for data in this column; 0 if the column is defined as FOR BIT DATA or is not a string type.
LOGGED	CHAR(1)	Applies only to columns whose type is LOB or distinct based on LOB; blank otherwise. <ul style="list-style-type: none"> <li>• "Y" = Column is logged</li> <li>• "N" = Column is not logged</li> </ul>
COMPACT	CHAR(1)	Applies only to columns whose type is LOB or distinct based on LOB; blank otherwise. <ul style="list-style-type: none"> <li>• "Y" = Column is compacted in storage</li> <li>• "N" = Column is not compacted</li> </ul>
INLINE_LENGTH	INTEGER	Maximum size in bytes of the internal representation of an instance of an XML document or a structured type that can be stored in the base table; 0 when not applicable.
IDENTITY	CHAR(1)	<ul style="list-style-type: none"> <li>• "Y" = Identity column</li> <li>• "N" = Not an identity column</li> </ul>
GENERATED	CHAR(1)	Type of generated column. <ul style="list-style-type: none"> <li>• "A" = Column value is always generated</li> <li>• "D" = Column values is generated by default</li> <li>• Blank = Column is not generated</li> </ul>

## **ADMINTEMPTABLES administrative view and ADMIN\_GET\_TEMP\_TABLES table function - Retrieve information for temporary tables**

The ADMINTEMPTABLES administrative view and the ADMIN\_GET\_TEMP\_TABLES table function provide methods to retrieve table attribute and instantiation time information for instances of created temporary tables and declared temporary tables.

Although the catalog views contain table attribute information for created temporary tables, they do not contain this information for declared temporary tables. In addition, the catalog views do not contain table instantiation time information for created temporary tables or declared temporary tables.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “ADMINTEMPTABLES administrative view”
- “ADMIN\_GET\_TEMP\_TABLES table function” on page 238

### **ADMINTEMPTABLES administrative view**

The ADMINTEMPTABLES administrative view returns table attribute and instantiation time information for instances of created temporary tables and declared temporary tables.

The schema is SYSIBMADM.

Refer to the Information returned for ADMINTEMPTABLES administrative view and ADMIN\_GET\_TEMP\_TABLES table for a complete list of information that can be returned.

### **Authorization**

One of the following authorizations is required:

- SELECT privilege on the ADMINTEMPTABLES administrative view
- CONTROL privilege on the ADMINTEMPTABLES administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the ADMIN\_GET\_TEMP\_TABLES table function
- DATAACCESS authority

### **Default PUBLIC privilege**

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

### **Examples**

*Example 1:* Retrieve table attributes and instantiation time information for all instances of created temporary tables and declared temporary tables currently existing in the database.

```
SELECT * FROM SYSIBMADM.ADMINTEMPTABLES
```

*Example 2:* Determine which connections have an instance of a created temporary table.

```
SELECT APPLICATION_HANDLE, TABSCHEMA, TABNAME
FROM SYSIBMADM.ADMINTEMPTABLES
WHERE TEMPTABTYPE = 'C'
```

*Example 3:* Retrieve table attributes and instantiation time information for all declared temporary tables declared for all the tables instantiated by the user that connected to the database.

```
SELECT TABSCHEMA, TABNAME, ONCOMMIT, ONROLLBACK,
INSTANTIATION_TIME
FROM SYSIBMADM.ADMINTEMPTABLES
WHERE TEMPTABTYPE = 'D' AND INSTANTIATOR = SYSTEM_USER
```

## ADMIN\_GET\_TEMP\_TABLES table function

The ADMIN\_GET\_TEMP\_TABLES table function returns the same information as the ADMINTABINFO administrative view, but allows you to specify a schema name and a table name.

Refer to the Information returned for ADMINTEMPTABLES administrative view and ADMIN\_GET\_TEMP\_TABLES table for a complete list of information that can be returned.

## Syntax

►►—ADMIN\_GET\_TEMP\_TABLES—(—*application\_handle*—,—*tabschema*—,—*tabname*—)————►◄

The schema is SYSPROC.

## Table function parameters

### *application\_handle*

An input argument of type BIGINT that specifies an application handle. If *application\_handle* is specified, data is returned for the specified connection only; if *application\_handle* is NULL, data is returned for all connections.

### *tabschema*

An input argument of type VARCHAR(128) that specifies a schema name.

### *tabname*

An input argument of type VARCHAR(128) that specifies a created temporary table name or a declared temporary table name.

## Authorization

One of the following authorizations is required:

- EXECUTE privilege on the ADMIN\_GET\_TEMP\_TABLES table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Examples

*Example 1:* Retrieve table attributes and instantiation time information for all instances of the created temporary table DBUSER1.EMPLOYEE for all connections.

```
SELECT TABSCHEMA, TABNAME, ONCOMMIT, ONROLLBACK, INSTANTIATION_TIME
FROM TABLE (SYSPROC.ADMIN_GET_TEMP_TABLES(NULL, 'DBUSER1', 'EMPLOYEE'))
AS T
```

*Example 2:* Retrieve the instantiation time and table space ID for all instances of user temporary tables for the current connection.

```
SELECT TABSCHEMA, TABNAME, INSTANTIATION_TIME, TBSP_ID
FROM TABLE (SYSPROC.ADMIN_GET_TEMP_TABLES(APPLICATION_ID(), '', ''))
AS T
```

## Usage notes

- 
- If both *tabschema* and *tabname* are specified, then information is returned for that specific temporary table only.
- If *tabschema* is specified but *tabname* is NULL or the empty string ("), then information is returned for all tables in the given schema.
- If *tabschema* is NULL or the empty string (") and *tabname* is specified, then an error is returned. To retrieve information for a specific temporary table, the table must be identified by both schema and table name.
- If both *tabschema* and *tabname* are NULL or the empty string ("), then information is returned for all temporary tables for the connection or all connections, depending on the value of *application\_handle*.
- If *tabschema* or *tabname* do not exist, or *tabname* does not correspond to a temporary table name, or instances of the identified temporary table do not exist in the database, then an empty result set is returned.

## Information returned

Table 42. Information returned for ADMINTEMPTABLES administrative view and ADMIN\_GET\_TEMP\_TABLES

Column name	Data type	Description
APPLICATION_HANDLE	BIGINT	application_handle - Application handle monitor element
APPLICATION_NAME	VARCHAR(256)	appl_name - Application name monitor element
TABSCHEMA	VARCHAR(128)	table_schema - Table schema name monitor element
TABNAME	VARCHAR(128)	table_name - Table name monitor element
INSTANTIATOR	VARCHAR(128)	Authorization ID under which the created temporary table was instantiated or declared temporary table was declared.
INSTANTIATOR_TYPE	CHAR(1)	<ul style="list-style-type: none"><li>• "U" = The instantiator is an individual user</li></ul>
TEMPTABTYPE	CHAR(1)	Temporary table type: <ul style="list-style-type: none"><li>• "C" = Created temporary table</li><li>• "D" = Declared temporary table</li></ul>

Table 42. Information returned for ADMINTEMPTABLES administrative view and ADMIN\_GET\_TEMP\_TABLES (continued)

Column name	Data type	Description
INSTANTIATION_TIME	TIMESTAMP	Time at which the created temporary table instance was instantiated or the declared temporary table was declared.
COLCOUNT	SMALLINT	Number of columns, including inherited columns (if any).
TAB_FILE_ID	BIGINT	table_file_id - The file ID (FID) for the table.
TBSP_ID	BIGINT	tablespace_id - An integer that uniquely represents a table space used by the current database.
PMAP_ID	SMALLINT	Identifier for the distribution map that is currently in use by this table.
PARTITION_MODE	CHAR(1)	Indicates how data is distributed among database partitions in a partitioned database system. <ul style="list-style-type: none"> <li>• "H" = Hashing</li> <li>• Blank = No database partitioning</li> </ul>
CODEPAGE	SMALLINT	Code page of the object. This is the default code page used for all character columns and expression-generated columns.
ONCOMMIT	CHAR(1)	Specifies the action taken on this table when a COMMIT operation is performed. <ul style="list-style-type: none"> <li>• "D" = Delete rows</li> <li>• "P" = Preserve rows</li> </ul>
ONROLLBACK	CHAR(1)	Specifies the action taken on this table when a ROLLBACK operation is performed. <ul style="list-style-type: none"> <li>• "D" = Delete rows</li> <li>• "P" = Preserve rows</li> </ul>
LOGGED	CHAR(1)	Specifies whether this table is logged. <ul style="list-style-type: none"> <li>• "N" = Not logged</li> <li>• "Y" = Logged</li> </ul>
TAB_ORGANIZATION	CHAR(1)	tab_organization - Data organization in table monitor element

### ADMIN\_IS\_INLINED function - Determine if data is inlined

The ADMIN\_IS\_INLINED function retrieves state information about inline data for an XML column, BLOB column, CLOB column, or DBCLOB column.

#### Syntax

►►—ADMIN\_IS\_INLINED—(—column-name—)—————►◄

The schema is SYSIBM.

#### Return value

This function returns one of the following values of type SMALLINT, or the null value:

- 1** Indicates that the data is inlined.
- 0** Indicates that the data is not inlined.

**NULL** Indicates that the inputs are NULL.

## Function parameters

### *column-name*

Identifies a column of the base table with a data type of XML, BLOB, CLOB, or DBCLOB (SQLSTATE 42884). The column must directly or indirectly reference the column of a base table that is not generated based on an expression (SQLSTATE 42815).

## Example

*Example 1:* The following example indicates whether the three XML documents in the XML column xml\_doc1 of the TAB1 table are inlined:

```
db2 => SELECT PK, ADMIN_IS_INLINED(xml_doc1) as IS_INLINED
 from TAB1
```

This query results in the following output:

PK	IS_INLINED
1	1
2	0
3	0

3 record(s) selected.

*Example 2:* The following example indicates whether one of the XML documents in the XML column xml\_doc1 of the TAB1 table is inlined:

```
db2 => SELECT PK, ADMIN_IS_INLINED(xml_doc1) as IS_INLINED
 from TAB1 where PK=1
```

This query results in the following output:

PK	IS_INLINED
1	1

1 record(s) selected.

*Example 3:* The following example indicates whether the three CLOB data contained in the CLOB column clob\_1 of the TAB1 table are inlined:

```
db2 => SELECT PK, ADMIN_IS_INLINED(clob_1) as IS_INLINED
 from TAB1
```

This query results in the following output:

PK	IS_INLINED
1	0
2	0
3	1

3 record(s) selected.

## ADMIN\_MOVE\_TABLE procedure - Move tables online

The ADMIN\_MOVE\_TABLE stored procedure moves the data in an active table into a new table object with the same name, while the data remains online and available for access.

This stored procedure creates a protocol table composed of rows that contain status information and configuration options related to the table to be moved. The return set from this procedure is the set of rows from the protocol table related to the table to be moved.

This stored procedure uses the following terminology:

**Source table**

The original table name that is passed in as a parameter into the stored procedure. This is the table to be moved.

**Target table**

A table created by the stored procedure using the table definition passed in through the stored procedure. All of the data from the source table is copied into this table and then it is renamed to the same name as the source table.

**Staging table**

A table created by the stored procedure. The staging table stores any update, delete or insert changes that occur on the source table during the execution of the table move. This table is dropped when the move is complete.

## Authorization

You must have SQLADM or DBADM authority to invoke the ADMIN\_MOVE\_TABLE stored procedure. You must also have the appropriate object creation authorities, including authorities to issue the SELECT statement on the source table, and to issue the INSERT statement on the target table.

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

## Syntax

There are two equally valid methods to start the ADMIN\_MOVE\_TABLE stored procedure:

- Use the first method to modify only certain parts of the table definition for the target table. For instance, if you had a table definition that is quite large (several KB), and all you want to do is modify the table spaces for the table, you can do so without having to determine the entire CREATE TABLE statement needed to recreate the source table. All you need to do is to fill out the **data\_tbsp**, **index\_tbsp**, and **lob\_tbsp** parameters, leaving the other optional parameters blank.
- The second method provides you with more control and flexibility. You can create the target table beforehand, rather than having the stored procedure create the target table. By creating the target table beforehand, you can create a target table that would not be possible using the first method.

Method 1:

```
➤➤ ADMIN_MOVE_TABLE (—tabschema—, —tabname—, —data_tbsp—, —index_tbsp—, —
```

► *lob\_tbsp*—, —*organize\_by\_clause*—, —*partkey\_cols*—, —*data\_part*—, —*coldef*—, —►

► *options*—, —*operation*—) —►

Method 2:

►► ADMIN\_MOVE\_TABLE—(—*tabschema*—, —*tabname*—, —*target\_tabname*—, —►

► *options*—, —*operation*—) —►

The schema for both methods is SYSPROC.

## Procedure parameters

### *tabschema*

This input parameter specifies the name of the schema which contains the table to be moved. This parameter is case sensitive and has a data type of VARCHAR(128). This parameter can be NULL or the empty string, if operation REPORT is used to monitor all table moves initiated.

### *tabname*

This input parameter specifies the name of the table to be moved. This parameter is case sensitive and has a data type of VARCHAR(128). This parameter can be NULL or the empty string, if operation REPORT is used to monitor all table moves initiated.

### *data\_tbsp*

This input parameter specifies the new data table space for the target table. If a value is provided, the *index\_tbsp* and *lob\_tbsp* parameters are required. If a value is not provided, the data table space of the source table is used. This parameter is case sensitive and has a data type of VARCHAR(128). This parameter can be NULL or the empty string.

### *index\_tbsp*

This input parameter specifies the new index table space for the target table. If a value is provided, the *data\_tbsp* and *lob\_tbsp* parameters are required. If a value is not provided, the index table space of the source table is used. This parameter is case sensitive and has a data type of VARCHAR(128). This parameter can be NULL or the empty string.

### *lob\_tbsp*

This input parameter specifies the new LOB table space for the target table. If a value is provided, the *data\_tbsp* and *index\_tbsp* parameters are required. If a value is not provided, the LOB table space of the source table is used. This parameter is case sensitive and has a data type of VARCHAR(128). This parameter can be NULL or the empty string.

### *organize\_by\_clause*

This input parameter can be used to specify an ORGANIZE BY clause for the table. If the value provided does not begin with 'ORGANIZE BY' then it

provides the multi-dimensional clustering (MDC) specification for the target table. The values are entered as a comma separated list of the columns used to cluster data in the target table along multiple dimensions. If a value of NULL or “-” is given, the ORGANIZE BY clause is not used. If an empty string or a single blank is given, the procedure checks whether there is an MDC or ITC specification on the source table, and uses that specification if located. If the argument begins with 'ORGANIZE BY' it can be used to specify any option related to the ORGANIZE BY clause of a CREATE TABLE statement. This parameter has a data type of VARCHAR(32672) and has the same format as the ORGANIZE BY DIMENSIONS clause of the CREATE TABLE statement. This parameter can be NULL, the empty string, or a single blank.

Example 1: 'C1, C4, (C3,C1), C2'

Example 2: ORGANIZE BY INSERT TIME

#### *partkey\_cols*

This input parameter provides the partitioning key columns specification for the target table. The values are entered as a comma separated list of the key columns that specify how the data is distributed across multiple database partitions. If a value of NULL or minus sign (-) is given, the PARTITIONING KEY clause is not used. If an empty string or a single blank is given, the procedure checks the existing distribution of the source table, and uses that specification, if located. This approach can be used to make the target table a random distribution table if the source table was created as a random distribution table. This parameter has a data type of VARCHAR(32672) and has the same format as the DISTRIBUTE BY HASH clause of the CREATE TABLE statement.

Example: 'C1, C3'

#### *data\_part*

This input parameter provides the data partitioning specification for the target table. This statement defines how to divide table data across multiple storage objects (called data partitions), according to the values in one or more of the table columns. If a value of NULL or minus sign (-) is given, the PARTITION BY RANGE clause is not used. If an empty string or a single blank is given, the procedure checks whether there is a data partition scheme on the source table, and uses that information (including partition name) if located. This parameter has a data type of VARCHAR(32672) and has the same format as the PARTITION BY RANGE clause of the CREATE TABLE statement.

Example: '(C1) (STARTING FROM (1) EXCLUSIVE ENDING AT (1000) EVERY (100))'

#### *coldef*

This input parameter specifies a new column definition for the target table, allowing you to change the column types as long as they are compatible; however, the column names must remain the same.

This also provides the ability to add new columns and drop existing columns. When adding a column, it must be defined as either nullable or have a default value set. Also, a column can only be dropped if there is a unique or primary index on the table and the column to be dropped is not a part of that unique or primary index. This parameter has a data type of VARCHAR(32672). This parameter can be NULL or the empty string.

Example: 'C1 INT, C2 INT DEFAULT 0'

### *target\_tabname*

This input parameter provides the name of an existing table to use as the target table during the move. The following changes can be made to the target table being passed in:

- The data, index and LOB table spaces can be changed
- The multi dimensional column (MDC) specification can be added or changed
- The partitioning key columns specification can be added or changed
- The data partitioning specification can be added or changed
- Data compression can be added or removed
- A new column definition can be specified; however the same restrictions as when specifying the *coldef* parameter apply here.

The following restrictions apply to the named table:

- The table must exist in the same schema as the source table
- The table must be empty
- No typed tables, materialized query tables (MQT), staging tables, remote tables or clustered tables are permitted

If this parameter is set to NULL or the empty string, the stored procedure uses the same definition as the source table. This parameter is case sensitive and has a data type of VARCHAR(128).

### *options*

This input parameter is a string comprised of comma separated values, which define any options used by the stored procedure. This list of options is not case sensitive and has a data type of VARCHAR(32672). The list value can be NULL or the empty string. The following values are valid:

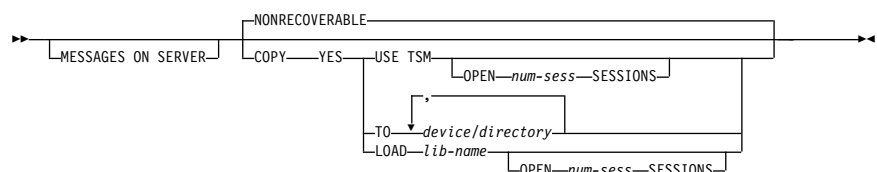
#### **ALLOW\_READ\_ACCESS**

This option allows the move of column-organized tables or convert column-organized tables to row-based tables. The source table can be still accessed using SELECT statements, but any UPDATE, DELETE or INSERT statement issued against the source table will cause a SQL error.

**KEEP** This option keeps a copy of the original source table under a different name. If the source table name is T1, then after the move that table will be automatically renamed to something such as T1AAAVxo. You can retrieve the exact name of the source table in the returned protocol table, under the ORIGINAL key. You may set this option at any point up to and including the SWAP phase.

#### **COPY\_USE\_LOAD “<load options>”**

##### **Syntax for COPY\_USE\_LOAD**



If you specify any load options for COPY\_USE\_LOAD, ADMIN\_MOVE\_TABLE uses an ADMIN\_CMD load to copy the data from the source table to the target table. If you do not specify any

options for COPY\_USE\_LOAD, then the NONRECOVERABLE option the db2Load API is used to copy the data from the source table to the target table.

#### **MESSAGES ON SERVER**

Specifies that the message file created on the server by the **LOAD** command is to be retained in case of load failures. The **WARNINGS** entry in the protocol table contain the message retrieval SQL statement that is required to retrieve all the warnings and error messages that occur during load, and the message removal SQL statement that is required to clean up the messages. Note that with or without the clause, the fenced user ID must have the authority to create files under the directory indicated by the **DB2\_UTIL\_MSGPATH** registry variable.

#### **COPY YES**

Specifies that a copy of the loaded data will be saved. This option is invalid if forward recovery is disabled.

#### **USE TSM**

Specifies that the copy will be stored using Tivoli Storage Manager (TSM).

#### **OPEN *num-sess* SESSIONS**

The number of I/O sessions to be used with TSM or the vendor product. The default value is 1.

#### **TO *device or directory***

Specifies the device or directory on which the copy image will be created.

#### **LOAD *lib-name***

The name of the shared library (DLL on Windows operating systems) containing the vendor backup and restore I/O functions to be used. It can contain the full path. If the full path is not given, it will default to the path where the user exit programs reside.

#### **NONRECOVERABLE**

Specifies that the load transaction is to be marked as nonrecoverable and that it will not be possible to recover it by a subsequent roll forward action. If COPY YES is not used, NONRECOVERABLE is the default.

#### **COPY\_WITH\_INDEXES**

This option creates indexes before copying the source table; however, the default is to create the indexes after copying the source table. The advantages of this option are that index creation after copying requires a whole table scan per index and that the index creation is a transaction that requires active log space. If the LOGINDEXBUILD database configuration parameter is on, significant log space is required for building the indexes in a short time frame. One disadvantage of this option is that copy performance is reduced because indexes need to be maintained on the target table. Also, the resulting indexes many contain *pseudo-deleted* keys, and the indexes are not as well balanced as if the indexes were created after the copy. You may set the COPY\_WITH\_INDEXES option at any point up to and including the COPY phase.

**FORCE**

If the force option is set, the SWAP phase does not check to see if the source table has changed its table definition.

**FORCE\_ALL**

Heavy workload on the source table might prevent ADMIN\_MOVE\_TABLE from completing the SWAP phase (SQL0911N reason code 2). In other words, if the source table is heavily used by other applications during the SWAP phase, ADMIN\_MOVE\_TABLE might rollback due to a deadlock (SQL0911N reason code 2).

Use this option to force all applications having workload on the source table. Forced applications will receive SQL1224N on all subsequent database manager requests.

Most of the time this option is not needed. Use FORCE\_ALL with care. You must have SYSADM, SYSCTRL or SYSMAINT authority to perform SWAP with FORCE\_ALL option.

**NO\_STATS**

This option does not start RUNSTATS or any statistic copying on the target table. If you use the AUTO\_RUNSTATS or AUTO\_STMT\_STATS database configuration parameters, the database system automatically creates new statistics afterward. For backwards compatibility, STATS\_NO is also accepted. You can set the NO\_STATS option at any point up to and including the SWAP phase.

**COPY\_STATS**

This option copies the statistics from the source table to the target table before performing the swap. This may cause inaccurate physical statistics, especially if the page size is changed. However, setting this option saves computing time as RUNSTATS is not called to compute new statistics. Also, the optimizer may choose the same access plans, because the statistics are the same. For backwards compatibility, STATS\_COPY is also accepted. You may set the STATS\_COPY option at any point up to and including the SWAP phase.

**NO\_AUTO\_REVAL**

This option prevents automatic revalidation on the table, and instead, re-creates all triggers and views. The NO\_AUTO\_REVAL option can be set only in the INIT phase.

**REORG**

This option sets up an extra offline REORG on the target table before performing the swap. If you use this option to improve your compression dictionary, be advised that using the default sampling approach is a better method to create an optimal compression dictionary. However, if you require an optimal XML compression dictionary, then REORG is the only method. You may set the REORG option at any point up to and including the SWAP phase.

**NO\_TARGET\_LOCKSIZE\_TABLE**

This option does not keep the LOCKSIZE table option on the target table during the COPY and SWAP phases. The default is to use the LOCKSIZE table option on the target table to prevent locking overhead, when no unique index is specified on the source table.

**CLUSTER**

This option reads the data from the source table with an ORDER BY clause when a copy index has been specified using

ADMIN\_MOVE\_TABLE\_UTIL, a clustering index exists on the source table or a unique index or primary key is defined in the source table.

**Note:** A copy index will override a clustering index; a clustering index will be used in preference to a primary key; a primary key will be used in preference to a unique index.

#### **NON\_CLUSTER**

This option reads the data from the source table without an ORDER BY clause regardless of whether a copy index has been specified, a clustering index exists on the source table, or a unique index or primary key has been defined in the source table.

**Note:** When neither CLUSTER or NON\_CLUSTER options are specified, ADMIN\_MOVE\_TABLE will read the data from the source table with an ORDER BY clause only when a clustering index exists on the source table.

#### **LOAD\_MSGPATH <path>**

This option can be used to define the load message file path.

LOAD\_MSGPATH can be used together with COPY\_USE\_LOAD in the following situations:

- COPY\_USE\_LOAD is specified without further options (in other words, the default COPY\_USE\_LOAD NONRECOVERABLE is assumed)
- COPY\_USE\_LOAD NONRECOVERABLE is specified explicitly

LOAD\_MSGPATH cannot be used together with COPY\_USE\_LOAD when optional keywords are also specified with COPY\_USE\_LOAD.

If the LOAD\_MSGPATH option is not specified, then the default path is determined by the DB2\_UTIL\_MSGPATH registry variable.

#### **LOAD\_TEMPPATH**

Use this option to define the load temporary file path.

#### **ALLOW\_RLBAC**

Use this option to move a table that has row-level label-based access control (RLBAC) security label specified. Before you use this option, ensure that you have adequate privilege; if you lack the privilege to access all the table row entries, data loss can occur.

#### **NOT\_ENFORCED**

Specify this option for the conversion of tables with enforced check constraints or foreign key (referential integrity) constraints that are not supported on the target table; otherwise, an error is returned (SQL1667N).

#### *operation*

This input parameter specifies which operation the stored procedure is to execute. There are two ways of calling the stored procedure: using the MOVE command to execute all the operations at one time; or by using the individual commands to execute the table move one step at a time. The main advantage of this second method is that you control when the SWAP phase actually occurs, thereby determining when the table is briefly taken offline. This allows you to make the move during a period of low system activity. If you use the individual commands, they must be called in the following order: INIT, COPY, REPLAY, VERIFY (optional), and SWAP.

- **MOVE:** Performs the entire table move (INIT, COPY, REPLAY, and SWAP operations) in one step.
- **INIT:** Verifies that a table move can take place, and initializes all of the data needed during the table move process (the target table, staging table, and the triggers on the source table).
- **COPY:** Copies the content from the source table to the target table. Any updates, deletes, or inserts occurring on the source table during this time are captured and stored in the staging table. New indexes are created at the end of the COPY phase, unless the COPY\_WITH\_INDEXES option is selected. Also, if needed, secondary indexes are created on the source and target tables to improve performance during the REPLAY phase. COPY can be used only after the INIT phase has completed.
- **REDIRECT:** Forwards changes directly to the target table instead of capturing the changes in the staging table.

**Note:** For tables with XML columns, the REDIRECT command does not work on multi-partitioned systems on tables that do not have a unique index.

- **REVERT:** Reverts to the original behavior wherein the staging table captures the changes.
- **REPLAY:** Copies into the target table any rows that have changed in the source table since the COPY phase began. REPLAY can be used only after the COPY phase has completed.
- **VERIFY:** Optionally checks if the table contents of the source and target tables are identical. This process involves obtaining a shared lock on the source and target tables, replaying any changes that have occurred on the source table, and then performing a comparison. If the table has a unique index, this command compares all values between columns that are in both tables. Otherwise, this command compares all values between columns that are in both tables (except for LONG, LOB or XML columns). This is an expensive operation and caution should be taken to decide if it is useful for your move. VERIFY can be used only after the COPY or REPLAY phases have completed.
- **SWAP:** Executes the REPLAY phase until the number of changes applied during the last scan of the staging table is less than the REPLAY\_THRESHOLD value stored in the protocol table. The source table is then taken offline briefly to finish the final REPLAY, and then this command swaps the source table with target table and brings the table back online. SWAP can be used after the COPY phase has completed, but ideally after the REPLAY phase has been called.
- **CLEANUP:** Drops the staging table, any non-unique indexes or triggers created on the source table by the stored procedure, and the source table if the KEEP option has not been set. CLEANUP can be called if the command failed during the SWAP phase.
- **CANCEL:** Cancels a multi-step table move while between phases, or cancels a failed table move operation. Executing this command requires that the operation status is not in COMPLETED or CLEANUP state. CANCEL clears up all intermediate data (the indexes, the staging table, the target table, and the triggers on the source table).
- **TERM:** Terminates a running or killed table move. TERM will kill a running table move, roll back all open transactions and set the table move to a well

defined operation status. From here, the table move can be canceled or continued. You must have SYSADM, SYSCTRL or SYSMANT authority to perform TERM.

- **REPORT:** Calculates a set of values to monitor the progress of a single or multiple table moves. Focus is the COPY and REPLAY phase of a running table move. To get values for all table moves, tabschema and tablename must be NULL or the empty string.

This parameter is not case sensitive and has a data type of VARCHAR(128).

### Information returned

*Table 43. Information returned by the ADMIN\_MOVE\_TABLE stored procedure*

Column name	Data type	Description
TABSCHEMA	VARCHAR(128)	table_schema - Table schema name monitor element
TABNAME	VARCHAR(128)	table_name - Table name monitor element
KEY	VARCHAR(32)	Name of the attribute.
VALUE	CLOB(10M)	Value of the attribute.

The key and value pairs that are returned in the result set can be found in Table 44. To modify the user configurable keys in the result set, use the ADMIN\_MOVE\_TABLE\_UTIL stored procedure.

*Table 44. Key and value pairs returned by the ADMIN\_MOVE\_TABLE stored procedure*

Key	Return Value	User Configurable
AGENT_ID	Displays the application_handle monitor element during the runtime of ADMIN_MOVE_TABLE.	No
APPLICATION_ID	Displays the appl_id monitor element during the runtime of ADMIN_MOVE_TABLE.	No
AUTHID	Displays the authorization ID of the user who called the stored procedure.	No
CANCEL_START	Displays the CANCEL phase start time.	No
CLEANUP_END	Displays the CLEANUP phase end time.	No
CLEANUP_START	Displays the CLEANUP phase start time.	No
COMMIT_AFTER_N_ROWS	During the COPY phase, a commit is executed after this many rows are copied. 0 means no commits during COPY. Default value is 10000.	Yes
COPY_ARRAY_SIZE	Specifies the ARRAY size for COPY_ARRAY_INSERT. A value less than or equal to 0 means do not use COPY_ARRAY_INSERT. Default value is 100.	Yes
COPY_END	Displays the COPY phase end time.	No
COPY_INDEXNAME	The name of the index used to cluster the data on the target table during the COPY phase. This value must be set before the COPY phase. The default name is the name of a cluster index on the source table, if it exists; otherwise the name of the unique or primary index on the source table.	Yes

Table 44. Key and value pairs returned by the ADMIN\_MOVE\_TABLE stored procedure (continued)

Key	Return Value	User Configurable
COPY_INDEXSCHEMA	The schema of the index used to cluster the data on the target table during the COPY phase. This value must be set before the COPY phase. The default schema is the schema name of a cluster index on the source table, if it exists; otherwise the schema name of the unique or primary index on the source table.	Yes
COPY_OPTS	The copy options used during the COPY phase.	No
COPY_START	Displays the COPY phase start time.	No
COPY_TOTAL_ROWS	Displays the total number of rows copied during the COPY phase.	No
DEEPCOMPRESSSION_SAMPLE	If the source table has compression enabled, this field specifies how much data (in KB) is sampled when creating a dictionary for compression. 0 means no sampling is done. Default value is 20MB (20480 KB).	Yes
INDEX_CREATION_TOTAL_TIME	Displays the total time required for creating secondary indexes.	No
INDEXNAME	Displays the name of the index or the empty string if the table does not have an index.	No
INDEXSCHEMA	Displays the schema of the index or the empty string if the table does not have an index.	No
INIT_END	Displays the INIT phase end time.	No
INIT_START	Displays the INIT phase start time.	No
LOCK	Displays the LOCK start time if another online table move stored procedure call is active, otherwise it is empty.	No
ORIGINAL	Displays the name of original table after the swap.	No
REORG_USE_TEMPSPACE	If you call the REORG option, you can also specify a temporary table space for the USE clause of the REORG command. If a value is not specified here, the REORG command uses the same table space as the table being reorganized.	Yes
REPLAY_END	Displays the REPLAY phase end time.	No
REPLAY_MAX_ERR_RETRIES	Specifies the maximum retry count for errors (lock timeouts or deadlocks) that may occur during the REPLAY phase. Default value is 100.	Yes
REPLAY_START	Displays the REPLAY phase start time.	No
REPLAY_THRESHOLD	For a single iteration of the REPLAY phase, if the number of rows applied to the staging table is less than this value, then REPLAY stops, even if new entries are made in the meantime. Default value is 100.	Yes
REPLAY_TOTAL_ROWS	Displays the accumulated number of replayed rows.	No
REPLAY_TOTAL_TIME	Displays the accumulated time in seconds used for replaying rows.	No
STAGING	Displays the name of the staging table.	No

Table 44. Key and value pairs returned by the ADMIN\_MOVE\_TABLE stored procedure (continued)

Key	Return Value	User Configurable
STATUS	Displays the current status of the online table move: <ul style="list-style-type: none"> <li>• INIT: INIT is in progress</li> <li>• COPY: COPY is in progress or is possible</li> <li>• REPLAY: REPLAY is in progress or REPLAY and SWAP are possible</li> <li>• CLEANUP: MOVE is complete, but cleanup has not finished or CLEANUP is possible</li> <li>• CANCEL: CANCEL is in progress, but cancel has not finished or CANCEL is possible</li> <li>• COMPLETE: MOVE and CLEANUP are complete</li> <li>• COMPLETE_WITH_WARNINGS: MOVE and CLEANUP are complete, however there are warnings (listed under the WARNINGS key).</li> </ul>	No
SWAP_END	Displays the SWAP phase end time.	No
SWAP_MAX_RETRIES	Specifies the maximum number of retries allowed during the SWAP phase (if lock timeouts or deadlocks occur). Default value is 10.	Yes
SWAP_RETRIES	Displays the number of retries performed during SWAP phase.	No
SWAP_START	Displays the SWAP phase start time.	No
TARGET	Displays the name of the target table.	No
TERM_END	Displays the TERM phase end time.	No
TERM_START	Displays the TERM phase start time.	No
UTILITY_INVOCATION_ID	Displays the unique identifier for the table move operation.	No
VERIFY_END	Displays the verification end time.	No
VERIFY_START	Displays the verification start time.	No
VERSION	Displays the version of the stored procedure.	No
WARNINGS	Displays warnings to pass on to the user. These warnings include: <ul style="list-style-type: none"> <li>• Revalidation of all failed objects</li> <li>• Control could not be granted to a user, group, or role</li> <li>• An index was not created because a column it references no longer exists</li> </ul>	No

If operation REPORT is used, monitoring values will be inserted into a Declared Global Temporary Table (DGTT) named SESSION.ADMIN\_MOVE\_TABLE.

Table 45. SESSION.ADMIN\_MOVE\_TABLE returned by the operation REPORT

Column name	Data type	Description
TABSCHEMA	VARCHAR(128)	Table schema of the table move to be reported. Value gets retrieved from protocol table.
TABNAME	VARCHAR(128)	Table name of the table move to be reported. Value gets retrieved from protocol table.

Table 45. *SESSION.ADMIN\_MOVE\_TABLE* returned by the operation *REPORT* (continued)

Column name	Data type	Description
STATUS	VARCHAR(22)	Status of the table move. Value gets retrieved from protocol table.
AGENT_ID	BIGINT	Agent ID of a running table move. Value gets retrieved from protocol table.
INIT_START	TIMESTAMP	Start time of INIT operation of the table move. Value gets retrieved from protocol table.
COPY_THROUGHPUT	BIGINT	COPY performance. <ul style="list-style-type: none"> <li>• Formula: <math>\text{COPY\_TOTAL\_ROWS} / (\text{MIN}(\text{CURRENT\_TIMESTAMP}, \text{COPY\_END}) - \text{COPY\_START})</math></li> <li>• Unit: rows/second</li> <li>• NULL, if calculation is not possible</li> </ul>
COPY_END	TIMESTAMP	Estimated COPY completion time. The value is calculated using COPY_THROUGHPUT, COPY_TOTAL_ROWS and CARD. <ul style="list-style-type: none"> <li>• NULL, if calculation is not possible or the COPY phase has ended.</li> </ul>
ROWS_STAGING	BIGINT	Number of rows in the staging table. Calculated using MON_GET_TABLE values rows_inserted - rows_deleted. <ul style="list-style-type: none"> <li>• Unit: rows</li> <li>• NULL, if calculation is not possible</li> </ul>
REPLAY_THROUGHPUT	BIGINT	REPLAY performance. <ul style="list-style-type: none"> <li>• Formula: <math>\text{REPLAYED\_TOTAL\_ROWS} / \text{REPLAY\_TOTAL\_TIME}</math></li> <li>• Unit: rows/second</li> <li>• NULL, if calculation is not possible</li> </ul>
INFLOW_STAGING	BIGINT	Number of rows per second that currently get inserted from staging table. <ul style="list-style-type: none"> <li>• Unit: rows/second</li> <li>• NULL, if calculation is not possible</li> </ul>
OUTFLOW_STAGING	BIGINT	Number of rows per second that currently get removed from staging table. <ul style="list-style-type: none"> <li>• Unit: rows/second</li> <li>• NULL, if calculation is not possible</li> </ul>
GROWTH_STAGING	BIGINT	Growth of staging table. <ul style="list-style-type: none"> <li>• Formula: <math>\text{OUTFLOW\_STAGING} - \text{INFLOW\_STAGING}</math></li> <li>• Unit: rows/second</li> <li>• NULL, if calculation is not possible</li> </ul>

Table 45. *SESSION.ADMIN\_MOVE\_TABLE* returned by the operation *REPORT* (continued)

Column name	Data type	Description
REPLAY_ECT	TIMESTAMP	Estimated REPLAY completion time. Special value 9999-12-31-23:59:59 indicates that under current progress and workload conditions, the move will not end. NULL, if calculation is not possible.
AVG_ROWS_PER_KEY	DOUBLE	The average number of rows, the replay key in staging table will touch. <ul style="list-style-type: none"> <li>• Formula: CARD/FULLKEYCARD</li> <li>• The input values for the formula get retrieved from SYSCAT.INDEXES and SYSCAT.TABLES</li> <li>• FULLKEYCARD gets retrieved for the replay index.</li> <li>• 1, if no statistics are available.</li> </ul>
CARD	BIGINT	The number of rows in the source table. Value gets retrieved from SYSCAT.TABLES

If operation *REPORT* is used, the result set is a subset of the DGTT and depends on reporting a single table move or all table moves initiated.

Result set to monitor a single table move:

- STATUS
- AGENT\_ID
- INIT\_START
- COPY\_THROUGHPUT
- COPY\_ECT
- ROWS\_STAGING
- REPLAY\_THROUGHPUT
- INFLOW\_STAGING
- OUTFLOW\_STAGING
- GROWTH\_STAGING
- REPLAY\_ECT

Result set to monitor all table moves initiated:

- TABSCHEMA
- TABNAME
- STATUS
- AGENT\_ID
- COPY\_ECT
- GROWTH\_STAGING

### Usage notes

**Suggestions for best results when using this procedure**

- Avoid making multiple moves into same table space at the same time. This prevents fragmentation on the target table space.
- Run this procedure when activity on the table is low. Avoid mass data loads or deletes so that parallel read access is not a problem.
- Use a multi-step move operation. The INIT and COPY phases can be called at any time. Execute the REPLAY phase multiple times in order to keep the staging table size small, and then issue the SWAP during a time of low activity on the table.
- Check if offline methods are a better choice for your table move, especially when considering tables without unique indexes and for tables with no index.

### Operations that are restricted on the source table

The stored procedure relies on triggers to capture any changes made to the source table. There are some operations that could affect the source table but which do not fire triggers. This could result in inconsistencies between the source and target table that cannot easily be detected by the stored procedures. These operations include:

- TRUNCATE TABLE (without restrict when delete triggers)
- IMPORT ... REPLACE INTO ...
- LOAD TABLE
- ALTER TABLE
- REORG (both online and offline)

These operations will be restricted on the source table using a new table-level state flag. The flag is set during the INIT phase and cleared during the CLEANUP or CANCEL phase. Restricted operations will fail with SQL0668N reason code 10 (sqlstate 57016).

### Operations that will affect the table move operation

There are operations that can cause the stored procedure to fail while a move is in progress. These operations include:

- Dropping the **SYSTOOLSPACE** table space
- Dropping/Renaming the source table
- Dropping/Renaming any of the temporary objects created by OTM in the INIT phase (target table, staging table, triggers on source table, protocol table)
- Altering values in the protocol table that are not listed as user configurable

### Naming convention for temporary objects

To avoid naming conflicts when creating temporary objects, the following naming convention is used:

- Postfix
  - “t” for target
  - “s” for staging
  - “o” for original
  - “g” for generated
  - “i” for insert trigger
  - “d” for delete trigger
  - “u” for before update trigger

- “v” for after update trigger
- Names are built consisting of <characters from name of object><base64 encoded hash key over name of object><postfix>.
- If length of name would exceed object length (128 bytes) <characters from name of object> gets shorter.
- Hash value gets calculated from the object name and is encoded similar to base64 encoding.

Sample:

```
Name of object: T1
Staging object: T1AAAVxs
Target object: T1AAAVxt
Original object: T1AAAVxo
Generated index: T1AAAVxg (if table has no index)
Insert trigger: T1AAAVxi
Delete trigger: T1AAAVxd
Before update trigger: T1AAAVxu
After update trigger: T1AAAVxv
```

### Online table move with compression and dictionary creation

There are several methods to create a data compression dictionary using Online Table Move. Compression must either be enabled on the source table or specified to be active in the new table definition if provided.

Create dictionary with sampling is the default method of Dictionary creation through Online Table Move. If compression is turned on for the table, then before performing the **COPY** operation, a system sampling of the data from the source table is inserted into the target table, where the amount of data sampled is specified in the DEEPCOMPRESSON\_SAMPLE field in the protocol table. The compression dictionary is then created based off of this random sample, and therefore results in an optimal compression dictionary.

Please note, that an XML compression dictionary will not be created through the sampling method. This is due to the fact that db2Inspect is used to create the compression dictionary, and db2Inspect currently does not have the ability to create an XML compression dictionary. The XML compression dictionary will be created through automatic dictionary creation (ADC).

Create dictionary with automatic dictionary creation (ADC) is the standard method of dictionary creation with tables. When you turn on compression for the table, the database system automatically creates the dictionary as data is inserted into the table. This results in a non-optimal compression dictionary. Note that the DEEPCOMPRESSON\_SAMPLE field in the protocol table must be set to 0 to avoid having the stored procedure attempt to create a better compression dictionary.

The create dictionary with REORG method of Dictionary creation results in a dictionary being created that reflects any activity on the source table that occurred while the COPY phase was in process. This is done by performing a REORG before the SWAP phase with the RESETDICTIONARY option set. An optimal dictionary will be created, however depending on the size of the table the REORG could take a long time. Also, if an optimal XML dictionary is required, REORG is the only method that will produce one. It is advised to use the sampling method of dictionary creation.

### Online table move and statistics on the table

If you use the **COPY\_USE\_LOAD** option and the source table has a statistics profile, the load operation collects statistics on the target table.

By default, when you are moving a table where statistics are collected, the **RUNSTATS** command is issued on the table during the SWAP phase. The **ADMIN\_MOVE\_TABLE** procedure attempts to create statistics only if the target table does not yet have any statistics. If the target table does not yet have statistics and you want to minimize the duration of the SWAP phase, issue the **RUNSTATS** command before the SWAP phase. If a statistics profile is found, the **RUNSTATS** command is called by using the statistics profile. Otherwise, the **RUNSTATS** command is called with the **WITH DISTRIBUTION ON COLUMNS (...)** and **AND SAMPLE DETAILED INDEXES ALL** clauses.

If the **COPY\_STATS** option has been set, the statistics from the source table are copied to the target table before performing the swap. Copying statistics may cause inaccurate physical statistics especially if changing page size. However, it will save on computing time as **RUNSTATS** does not have to be called to compute new statistics. Also, the optimizer may choose the same access plans, because the statistics are the same (plan stability). The statistics that are copied are in the **SYSSTAT.TABLES**, **SYSSTAT.COLUMNS**, **SYSSTAT.COLDIST**, **SYSSTAT.INDEXES**, and **SYSSTAT.COLGROUPS** catalog views.

If the **NO\_STATS** option has been set, the **RUNSTATS** command is not issued, and statistics are not copied on the target table. If you use **AUTO\_RUNSTATS** or **AUTO\_STMT\_STATS**, the database system automatically generates new statistics

#### Online table move with **LOAD** used for **COPY**

If you are using the **COPY\_USE\_LOAD** option, and if you do not specify a sub-option or you choose **NONRECOVERABLE**, then it is necessary to perform a backup of the target table space or table spaces before the SWAP phase in order to ensure recoverability. A backup can be created by issuing a statement such as the following:

```
BACKUP DB dbname TABLESPACE targetDataTablesapce,
 targetIndexTablespace, targetLongTablespace
ONLINE TO <destination>
```

#### Online table move with recoverable **LOAD** in HADR environment

If the destination for a recoverable **LOAD** in an HADR environment cannot be found from the standby, the table space will be inaccessible. The primary is not informed about this situation, so you might assume that the standby is up to date, but if there is a failover to the standby the table will not be accessible.

#### Online table move with generated columns

The Table Move stored procedure treats any generated columns in the source table specially. The following paragraphs describe how the different types of generated columns are handled.

A **row change timestamp column** is a column that holds a timestamp representing the time when a row was last changed.

If a row change timestamp column is found in the source table, the values of this column after the table move operation is complete will not be the same as they were before the table move operation. The values of the column after the table move will represent the time at which the rows were inserted/updated in the new table object. This is done because the actual rows are being changed and the row change timestamp column

values should therefore reflect these changes. If you use `ALLOW_READ_ACCESS` and `COPY_USE_LOAD` option of `ADMIN_MOVE_TABLE`, the row change timestamp column content of source table gets preserved during the move.

If a new table definition is supplied, and a column is defined as a row change timestamp column in the source table but not in the new table definition, then the column will not be a row change timestamp column.

An **identity column** is a column that automatically generates a value for the column when a row is inserted into the table.

If an identity column is found in the source table, the values of this column after the Table Move operation is complete will be identical to the values that were present before the table move operation. However, there is no way to determine the “last/next” value for the identity column in the source table. Therefore, when creating the identity column on the target table the value generation will be set to begin from the next “uncached” value. This is the same behavior that happens when the database restarts (stop/start). This behavior is documented in “ALTER TABLE statement” in the *SQL Reference Volume 2*, under the “SET NO CACHE or CACHE integer-constant” heading of the “identity-alteration” section.

The column will initially be created as a regular column in the target table, and then be altered to be an identity column during the brief offline period of the SWAP phase. This is done because the column may have been created as `GENERATED ALWAYS`, and that would block the stored procedure from being able to insert the exact values from the source table into the column in the target table.

If a new table definition is specified, and a column is specified to be an identity column in the new table definition, then the stored procedure will check to see if the definition of the identity column matches the definition of the column in the source table. If they are a match, the stored procedure will continue as previously described. If they are not a match, the stored procedure will use the new identity column definition. Please note that this will restart the identity column counter with whatever the start value is specified as, however the current values of the rows in the column will remain the same.

If a new table definition is specified, and a column that is specified as an identity column in the source table is not specified as an identity column in the new table definition, then the stored procedure will still create the column as an identity column in the target table using the same specification found in the source table. This is done so that users do not need to look up the definition of the existing identity column and re-enter it into the new table definition. If the user does not want to keep the column as an Identity column, then they can alter the target table after the call to the stored procedure to remove the identity specification from the column.

An **expression column** is a column that automatically generates a value for the column based on an expression when a row is inserted into the table.

If an expression column is found in the source table, the values of this column after the Table move operation is complete will be identical to the values that were present before the table move operation.

The column will originally be created as a regular column in the target table, and then be altered to be an expression column during the brief

offline period of the SWAP phase. This is done because expression columns are created as GENERATED ALWAYS, and do not allow inserts into that column. However, In order to alter the column in the target table to be an expression column, set integrity will briefly be turned off on the target table. The ALTER statement is performed, and then integrity is set back on with the GENERATED COLUMN IMMEDIATE UNCHECKED option.

The stored procedure will not support column expressions that include the table name (such as table 'T1' with expression (T1.C \*5)) in either the source table or the target table. To remedy this, the user can alter the column to change the expression to not include the table name.

If a new table definition is specified, and a column is specified to be an expression column in the new table definition, then the stored procedure will check to see if the definition of the expression column matches the definition of the column in the source table by performing a basic string to string comparison. If they are a match, the stored procedure will continue as previously described. If they are not a match, the stored procedure will use the new expression column definition. Please note that the current values of the rows in the column will remain the same.

If a new table definition is specified, and a column that is specified as an expression column in the source table is not specified as an expression column in the new table definition, then the stored procedure will still create the column as an expression column in the target table using the same specification found in the source table. This is done so that users do not need to look up the definition of the existing expression column and re-enter it into the new table definition. If the user does not want to keep the column as an Expression column, then they can alter the target table after the call to the stored procedure to remove the Expression Specification from the column.

### Online table move and objects and privileges that are preserved

The stored procedure will preserve the following objects when a Table Move is performed:

**Views** During the brief offline period during the SWAP phase, the views are dropped from the source table and are re-created on the target table.

Transfer of ownership is also performed to change the ownership of the view back to the original owner.

The granted privileges for the views are preserved only if the **auto\_reval** configuration parameter is set to DEFERRED or IMMEDIATE.

#### Triggers

During the brief offline period during the SWAP phase, the triggers are dropped from the source table and are re-created on the target table.

Transfer of ownership is also performed to change the ownership of the trigger back to the original owner.

#### Indexes

Indexes are created onto the target table at several times during the table move procedure. Indexes are first created at the end of the COPY phase, unless the **COPY\_WITH\_INDEXES** option is set then the indexes are first created at the beginning of the COPY phase. The

store procedure then looks for any newly created indexes, judging by index name alone, at the beginning of the REPLAY and SWAP phases. If new indexes are found, they are created. However, the stored procedure does look to see if any indexes have been deleted on the source table.

The index names are the same as they were on the source table for user created indexes. However, system created indexes can not be guaranteed to have the same name.

The indexes that are preserved are of the following type: 'REG', 'CLUST', and 'XVIL'.

Any user created indexes that reference a column that is being dropped in the target table are not preserved.

When moving from a source partitioned table to a target partitioned table, the partitioned attribute of the index are preserved. When moving from a source partitioned table to a target non-partitioned table, or vice-versa, the partitioned attribute is decided by the default behavior of the database.

### Constraints

Constraints (other than referential constraints) are re-created on the target table using the same constraint names. However, for unique and primary constraints the underlying index name may be different than the index name on the source table.

### Table flags

The table flags of the source table are created on the target table as soon as the target table is created in the INIT phase. These flags are as follows: 'append\_mode', 'locksize', 'volatile', 'compression', 'datacapture', 'pctfree', 'logindexbuild', 'owner', and 'droprule'. These flags are then checked at the end of the COPY phase and during the SWAP phase. If there are any changes in the flags, they are updated in the target table.

To keep the database recoverable and compatible with HADR setups, ADMIN\_MOVE\_TABLE does not copy the NOT LOGGED INITIALLY information from the source to the target table.

### Grant/Revoke

During the SWAP phase, the stored procedure goes through the entries in **SYSCAT.TABAUTH** and reproduces the granting of privileges on the table to users/groups/roles.

If the caller of the stored procedure does not have ACCESSCTRL or SECADM authority, then the CONTROL privilege cannot be granted. A list of all users/groups/roles that were not granted the CONTROL privilege can be found in the protocol table where the key is WARNINGS.

### Usage lists

During the brief offline period during the SWAP phase, usage lists defined on the source table or on the source table indexes are dropped and re-created on the target table. Any usage list that was in the active state before the move is re-activated after the move.

If auto\_revalidation is enabled on the database, and the **USE\_AUTO\_REVAL** option is not set (which is the default if auto\_revalidation is enabled), then the views are not be dropped as outlined previously. Instead, the views

remain and are revalidated with `auto_revalidation`. Triggers are dropped and re-created by the stored procedure because there is currently a limitation with renaming a table with a trigger defined as the subject. There might also be invalid views after the stored procedure completes if the **`auto_reval`** configuration parameter is set to `DEFERRED`. Objects are revalidated automatically on the next access, or you can call the `ADMIN_REVALIDATE_DB_OBJECTS` to revalidate all objects in the database.

### **Online table move with clustering over an index**

It is possible to cluster the target table by an index. If a cluster index is present on the source table, it will be clustered by that index by default. The default can be changed after the `INIT` phase (This implies phase wise execution of Online Table Move). Calling Online Table Move in one `MOVE` phase with no cluster index present will result in the stored procedure clustering the target table with the unique/primary index. If a cluster index exists, the stored procedure will cluster the target table using the cluster index.

If there is a cluster index on the source table, it is possible to not cluster the target table on the cluster index by performing a multi-step move and deleting the key entries `COPY_INDEXSCHEMA` and `COPY_INDEXNAME` from the protocol table after the `INIT` phase.

It is possible to cluster the target table by any secondary index by performing a multi-step move and inserting/updating the key entries `COPY_INDEXSCHEMA` and `COPY_INDEXNAME` in the protocol table with the required index to cluster the target table.

### **Online table move in a Db2 pureScale environment**

In a Db2 pureScale environment, concurrent read access is the default behavior. Concurrent write access is not allowed during index creation. Perform one of the following actions before you run the `ADMIN_MOVE_TABLE` procedure in a Db2 pureScale environment:

- If an index is not defined on the source table, create an index manually before you move the table
- If an index is defined on the source table, run the `ADMIN_MOVE_TABLE` procedure with the `COPY_WITH_INDEXES` option

### **Changing index attributes**

If a user wants to modify the attributes of any existing attributes (such as index clustering, index compression, change global to local indexes and vice versa) they can manually make these changes during a multi-step move operation.

This can be done by performing the `INIT` and `COPY` phases of the move via a multi-step move. Then manually make any changes to the indexes on the target table. The name of the target table can be found in the protocol table. After the modifications have finished, resume with the `REPLAY` and `SWAP` phases.

### **Using the `ADMIN_MOVE_TABLE` procedure to convert row-organized tables into column-organized tables**

You can convert the tables in either of the following ways:

- By specifying a column-organized target table

- By specifying the `ORGANIZE BY COLUMN` clause for the *organize\_by\_clause* parameter.

The `ADMIN_MOVE_TABLE` stored procedure remains online.

If you convert a row-organized table into a column-organized table, applicable column-organized table restrictions on queries (that is, limited isolation levels) take effect at the end of processing, after the new table becomes visible to queries.

The `ADMIN_MOVE_TABLE` procedure requires triggers on the source table to capture changes. However, because this type of trigger is not yet supported on column-organized tables, the move of column-organized tables can be performed only when you use the `ALLOW_READ_ACCESS` option. By using the `ALLOW_READ_ACCESS` option, the delete and insert operations are not allowed on the source table, while the move is in progress.

Prior to introducing this secondary index support on column-organized tables, where possible the `ADMIN_MOVE_TABLE` procedure moved primary and unique constraints and silently converted unique indexes into unique constraints, ignoring options that were not supported. All non-unique indexes were ignored.

With secondary indexes supported, the behavior will be that unique indexes and non unique indexes will be moved with the table where possible. The following index features that are not supported on column organized tables will be ignored and the index will be moved without them:

- **RANDOM** key order
- **CLUSTER** clause
- **INCLUDE** columns

Other unsupported features will result in the move failing with *sqlcodes* such as SQL2103N or SQL1667N.

The `PARTITION BY RANGE` clause is not supported for column-organized tables. To ignore the `PARTITION BY RANGE` clause, specify the minus sign (-) for the *data\_part* parameter.

MDC or ITC tables are converted into column-organized tables when you specify the `ORGANIZE BY COLUMN` clause for the *organize\_by\_clause* parameter.

If you use the load utility to populate a column-organized target table, the `ADMIN_MOVE_TABLE` procedure inherits column-organized table restrictions that apply to the load utility.

The `REORG` option is not supported.

A long offline `SWAP` phase occurs when the table has foreign key (referential integrity) constraints that are defined on it, because the foreign key relationship must be checked.

## Restrictions

The following restrictions apply to the `ADMIN_MOVE_TABLE` stored procedure:

- Only simple tables are supported as the source table. No materialized query tables, typed tables, range clustered tables, system tables, views, nicknames, or aliases are permitted.

- A table cannot be moved if there is an index with an expression-based key defined on the table.
- A table cannot be moved if an event monitor is currently active on the table.
- Tables without a unique index are subject to a complex and potentially expensive replay phase.
- A generated column cannot be part of the MDC specification.
- Be aware of the large disk space requirements, as the procedure creates two copies of the table and indexes, plus a staging table and log space.
- Copy performance may be an issue as most of the data is moved to the new table using “insert from select” form.
- The VERIFY operation for tables without a unique index does not work on tables with LOBs.
- The SYSTOOLSPACE table space must be created and accessible to 'PUBLIC'.
- Lock timeouts are possible during the COPY phase because of long running transactions on the source table.
- Deadlocks can occur during the SWAP phase.
- Deadlocks can occur on a source table with non-unique indexes and several update processes.
- With VARCHAR2 support enabled, the database treats the empty string and NULL as equivalent values, but the single blank is a distinct value. With VARCHAR2 support enabled, the *mdc\_cols*, *partkey\_cols*, and *data\_part* parameters can use a single blank as distinct from the empty string and NULL.
- A table cannot be moved if it is in the Set Integrity Pending state.
- A table cannot be moved if there are any XSR objects dependent on it.
- CHECK constraints with three part names are not supported and will return SQL0205. Use only the column name instead.

## Examples

1. This example calls the stored procedure using the first method, where the target table is defined within the procedure, to move a table named T1 which is located in the schema SVALENTI.

```
CALL SYSPROC.ADMIN_MOVE_TABLE(
'SVALENTI',
'T1',
'ACCOUNTING',
'ACCOUNT_IDX',
'ACCOUNT_LONG',
'',
'',
'',
'CUSTOMER VARCHAR(80), REGION CHAR(5), YEAR INTEGER, CONTENTS CLOB',
'',
'MOVE')
```

The following is an example of output from this query

Result set 1

-----

KEY	VALUE
-----	-----
AUTHID	SVALENTI
CLEANUP_END	2009-02-13-11.34.07.609575
CLEANUP_START	2009-02-13-11.34.07.369331
COPY_END	2009-02-13-11.34.05.148018
COPY_OPTS	BY_KEY,OVER_INDEX
COPY_START	2009-02-13-11.34.04.841292

```

COPY_TOTAL_ROWS 100
INDEXNAME T1_INDEX
INDEXSCHEMA SVALENTI
INDEX_CREATION_TOTAL_TIME 0
INIT_END 2009-02-13-11.34.04.552875
INIT_START 2009-02-13-11.34.03.013563
PAR_COLDEF CUSTOMER VARCHAR(80), REGION CHAR(5),
 YEAR INTEGER, CONTENTS CLOB
REPLAY_END 2009-02-13-11.34.06.198369
REPLAY_START 2009-02-13-11.34.05.164582
REPLAY_TOTAL_ROWS 100
REPLAY_TOTAL_TIME 5
STATUS COMPLETE
SWAP_END 2009-02-12-11.34.07.214447
SWAP_RETRIES 0
SWAP_START 2009-02-13-11.34.06.244506
VERSION 09.07.0000

```

22 record(s) selected.

Return Status = 0

2. This example calls the stored procedure using the second method, where the target table is created outside the procedure and is then named within the *target\_tabname* parameter, to move the same table as in the previous example.

The first step is to create the table manually:

```

CREATE TABLE SVALENTI.T1_TARGET (
 CUSTOMER VARCHAR(80),
 REGION CHAR(5),
 YEAR INTEGER,
 CONTENTS CLOB)
IN ACCOUNTING
INDEX IN ACCOUNT_IDX
LONG IN ACCOUNT_LONG'

```

Then call the stored procedure and pass in the name of the target table:

```

CALL SYSPROC.ADMIN_MOVE_TABLE(
'SVALENTI',
'T1',
'T1_TARGET',
'',
'MOVE')

```

The following is an example of output from this query

Result set 1

-----

KEY	VALUE
-----	-----
AUTHID	SVALENTI
CLEANUP_END	2009-02-13-11.37.49.283090
CLEANUP_START	2009-02-13-11.37.49.125786
COPY_END	2009-02-13-11.37.47.806060
COPY_OPTS	BY_KEY,OVER_INDEX
COPY_START	2009-02-13-11.37.47.446616
COPY_TOTAL_ROWS	0
INDEXNAME	T1_INDEX
INDEXSCHEMA	SVALENTI
INDEX_CREATION_TOTAL_TIME	1
INIT_END	2009-02-13-11.37.47.287703
INIT_START	2009-02-13-11.37.46.052952
PAR_COLDEF	using a supplied target table so COLDEF could be different
REPLAY_END	2009-02-13-11.37.48.785503
REPLAY_START	2009-02-13-11.37.47.822109
REPLAY_TOTAL_ROWS	0

```

REPLAY_TOTAL_TIME 0
STATUS COMPLETE
SWAP_END 2009-02-13-11.37.48.977745
SWAP_RETRIES 0
SWAP_START 2009-02-13-11.37.48.825228
VERSION 09.07.0000
22 record(s) selected.

```

Return Status = 0

3. In this example, the ADMIN\_MOVE\_TABLE stored procedure is used to convert the row-organized STAFF table into a column-organized table. This example specifies the existing STAFF2 table as the target table.

```

CALL SYSPROC.ADMIN_MOVE_TABLE(
 'OTM01COL',
 'STAFF',
 'STAFF2',
 'COPY_USE_LOAD',
 'MOVE'
)

```

4. In this example, the ADMIN\_MOVE\_TABLE stored procedure is used to convert the row-organized STAFF table into a column-organized table without specifying a target table. The ORGANIZE BY COLUMN clause is specified as a parameter so that the target table is created as a column-organized table.

```

CALL SYSPROC.ADMIN_MOVE_TABLE(
 'OTM01COL',
 'STAFF',
 '',
 '',
 '',
 'ORGANIZE BY COLUMN',
 '',
 '',
 '',
 'COPY_USE_LOAD',
 'MOVE'
)

```

5. Use the ADMIN\_MOVE\_TABLE stored procedure to convert the row-organized ACT table into a column-organized table. The ACT table has been altered to define an enforced foreign key constraint, as shown in the following example:

```

ALTER TABLE "TANJINXU"."ACT"
 ADD CONSTRAINT "RPAA" FOREIGN KEY ("ACTNO")
 REFERENCES "TANJINXU"."ACT" ("ACTNO")
 ON DELETE RESTRICT
 ON UPDATE NO ACTION
 ENFORCED
 ENABLE QUERY OPTIMIZATION

```

Because enforced foreign key (referential integrity) constraints are not supported on column-organized tables, the NOT\_ENFORCED option must be specified so that the target table can be created as a column-organized table.

```

CALL ADMIN_MOVE_TABLE(
 'TANJINXU',
 'ACT',
 '',
 '',
 'ORGANIZE BY COLUMN',
 '',
 ''
)

```

```

',
'COPY_USE_LOAD,NOT_ENFORCED',
'MOVE,TRACE'
)

```

6. Check the status of a table move for a table named *mytable* by using the following query:

```
SELECT * FROM SYSTOOLS.ADMIN_MOVE_TABLE WHERE KEY ='STATUS' AND TABNAME='mytable'
```


If the status is not COMPLETED, you can call the stored procedure again, specifying the appropriate option:

- If the status of the procedure is INIT, use the INIT operation, followed by other steps.
- If the status of the procedure is COPY, use the COPY operation, followed by other steps.
- If the status of the procedure is REPLAY, use the REPLAY or SWAP option.
- If the status of the procedure is CLEANUP, use the CLEANUP option.

If the status of an online table move is not COMPLETED or CLEANUP, you can call the stored procedure again, specifying the CANCEL option.

If the status is not COMPLETED and the MOVE operation was specified, CANCEL the operation and start over, because the MOVE operation runs INIT, COPY, REPLAY, and SWAP operations in one step.

#### Related information:

 [Technical article: Convert row-organized tables to column-organized tables in Db2 with BLU Acceleration](#)

### ADMIN\_MOVE\_TABLE\_UTIL procedure - Modify the online move table procedure

The ADMIN\_MOVE\_TABLE\_UTIL procedure works in conjunction with the SYSPROC.ADMIN\_MOVE\_TABLE stored procedure when moving active table data.

This stored procedure provides a mechanism to alter the user definable values in the ADMIN\_MOVE\_TABLE protocol table, which is created and used by the ADMIN\_MOVE\_TABLE procedure.

This procedure will only modify a value in the ADMIN\_MOVE\_TABLE protocol table if a table move for the table referenced by the TABSCHEMA and TABNAME parameters is already in progress, and the authorization ID of the caller of the procedure is the same as the user executing the table move.

#### Authorization

One of the following authorizations is required to use the routine:

- EXECUTE privilege on the ADMIN\_MOVE\_TABLE\_UTIL procedure
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, the authorization ID used must be the same as the one used to call the ADMIN\_MOVE\_TABLE stored procedure.

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

## Syntax

```
►►—ADMIN_MOVE_TABLE_UTIL—(—tabschema—,—tabname—,—action—,—key—,—value—)————►◄
```

The schema for this stored procedure is SYSPROC.

## Procedure parameters

### *tabschema*

This input parameter specifies the name of the schema containing the table being moved. This name is case sensitive. and has a data type of VARCHAR(128).

### *tabname*

This input parameter specifies the name of the table being moved. This parameter is case sensitive and has a data type of VARCHAR(128)

### *action*

This input parameter specifies the action for the procedure to execute.

Valid values are:

- UPSERT: If the specified TABSCHEMA.TABNAME.KEY exists in the ADMIN\_MOVE\_TABLE protocol table, this updates the corresponding VALUE with the new *value* parameter. Otherwise, this inserts the KEY and VALUE pair into the ADMIN\_MOVE\_TABLE protocol table.
- DELETE: If the specified TABSCHEMA.TABNAME. KEY exists in the ADMIN\_MOVE\_TABLE protocol table, this deletes the specified KEY and VALUE pair from the ADMIN\_MOVE\_TABLE protocol table.

This parameter has a datatype of VARCHAR(128).

### *key*

This input parameter specifies the key that to "upsert" or delete in the ADMIN\_MOVE\_TABLE protocol table.

Valid values are:

- COMMIT\_AFTER\_N\_ROWS: During the COPY phase, a commit is executed after this many rows are copied. A value of 0 means no commits are executed during COPY.
- DEEPCOMPRESSSION\_SAMPLE: If the source table has compression enabled, this field specifies how much data (in KB) is sampled when creating a dictionary for compression. A value of 0 means no sampling is done.
- COPY\_ARRAY\_SIZE: Specifies the ARRAY size for COPY\_ARRAY\_INSERT, a value less than or equal to 0 means do not use COPY\_ARRAY\_INSERT.
- COPY\_INDEXSCHEMA: The schema of the index used to cluster the data on the target table during the COPY phase.
- COPY\_INDEXNAME: The name of the index used to cluster the data on the target table during the COPY phase.
- REPLAY\_MAX\_ERR\_RETRIES: Specifies the maximum retry count for errors (lock timeouts or deadlocks) that may occur during the REPLAY phase.

- **REPLAY\_THRESHOLD:** For a single iteration of the REPLAY phase, if the number of rows applied to the staging table is less than this value, then REPLAY stops, even if new entries are made in the meantime.
- **REORG\_USE\_TEMPSPACE:** If you call the REORG option in the table move, you can also specify a temporary table space for the USE clause of the REORG command. If a value is not specified here, the REORG command uses the same table space as the table being reorganized.
- **SWAP\_MAX\_RETRIES:** Specifies the maximum number of retries allowed during the SWAP phase (if lock timeouts or deadlocks occur).

This parameter has a data type of VARCHAR(128).

#### *value*

This input parameter specifies the value to "upsert" into the ADMIN\_MOVE\_TABLE protocol table. This parameter has a data type of CLOB(10M). The parameter can be NULL or the empty string.

### Usage notes

More information regarding the changeable KEY values in the ADMIN\_MOVE\_TABLE protocol table is available in the Usage notes section of the ADMIN\_MOVE\_TABLE procedure.

### Examples

This example covers a basic call to the stored procedure in order to update the compression value and remove the specific index information used for the target table copying.

First, the ADMIN\_MOVE\_TABLE procedure is called to start the table move process before calling the ADMIN\_MOVE\_TABLE\_UTIL procedure in order to update or delete a value in the ADMIN\_MOVE\_TABLE protocol table:

```
CALL SYSPROC.ADMIN_MOVE_TABLE('SVALENTI','T1','','','','','','','','','INIT')
```

Next, update the DEEP\_COMPRESSION\_SAMPLE value to 30720 KB:

```
CALL SYSPROC.ADMIN_MOVE_TABLE_UTIL('SVALENTI','T1','UPsert',
 'DEEPCompression_SAMPLE','30720')
```

Now, delete the COPY\_INDEXSCHEMA and COPY\_INDEXNAME values:

```
CALL SYSPROC.ADMIN_MOVE_TABLE_UTIL('SVALENTI','T1','DELETE','COPY_INDEXSCHEMA','')
CALL SYSPROC.ADMIN_MOVE_TABLE_UTIL('SVALENTI','T1','DELETE','COPY_INDEXNAME','')
```

After these changes, continue the ADMIN\_MOVE\_TABLE procedure using the new values in the meta table:

```
CALL SYSPROC.ADMIN_MOVE_TABLE('SVALENTI','T1','','','','','','','','','COPY')
CALL SYSPROC.ADMIN_MOVE_TABLE('SVALENTI','T1','','','','','','','','','REPLAY')
CALL SYSPROC.ADMIN_MOVE_TABLE('SVALENTI','T1','','','','','','','','','SWAP')
```

### **ADMIN\_REMOVE\_MSGS procedure - Clean up messages generated by a data movement utility that is executed through the ADMIN\_CMD procedure**

The ADMIN\_REMOVE\_MSGS procedure is used to clean up messages generated by a single execution of a data movement utility command through the ADMIN\_CMD procedure.

The input parameter *operation\_id* identifies the operation.

## Syntax

►—ADMIN\_REMOVE\_MSGS—(*operation\_id*)—►

The schema is SYSPROC.

## Procedure parameter

### *operation\_id*

An input argument of type VARCHAR(139) that specifies the operation ID of the message file(s) produced by a data movement utility that was executed through the ADMIN\_CMD procedure. The operation ID is generated by the ADMIN\_CMD procedure.

## Authorization

EXECUTE privilege on the ADMIN\_REMOVE\_MSGS procedure. The fenced user ID must be able to delete files under the directory indicated by registry variable **DB2\_UTIL\_MSGPATH**. If the registry variable is not set, then the fenced user ID must be able to delete the files in the tmp subdirectory of the instance directory.

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

## Example

Clean up messages with operation ID '24523\_THERESAX'.

```
CALL SYSPROC.ADMIN_REMOVE_MSGS('24523_THERESAX')
```

## Usage notes

The CALL statement that invokes this procedure with the appropriate *operation\_id* can be found in the MSG\_REMOVAL column of the first result set returned by ADMIN\_CMD procedure.

## ADMIN\_REVALIDATE\_DB\_OBJECTS procedure - Revalidate invalid database objects

The ADMIN\_REVALIDATE\_DB\_OBJECTS procedure revalidates database objects.

This procedure takes various input parameters that control the level of revalidation that is performed:

- To revalidate all of the invalid objects in the database, either specify NULL for all parameters, or call the procedure without parameters.
- To revalidate all of the invalid database objects under a specific schema, specify a value for *object\_schema*, and specify NULL for *object\_name* and *object\_type*.
- To revalidate a specific invalid database object, specify valid values for the first parameters.
- To force the revalidation of compiled SQL PL or PL/SQL objects regardless of their state being valid or invalid, specify a value for *object\_type*, *object\_schema*, *object\_name*, and specify 'Y' for *force*.
- To enable or disable debug on compiled SQL PL or PL/SQL objects being revalidated, specify 'Y' or 'N' for *debug\_flag*.

## Syntax

```
►—ADMIN_REVALIDATE_DB_OBJECTS—(—object_type—,—object_schema—,——————►
►—object_name—,—force—,—debug_flag—)——————►◀
```

The schema is SYSPROC.

## Procedure parameters

### *object\_type*

An input argument of type VARCHAR(30) that identifies the type of the database object. The following types are valid:

- FUNCTION
- GLOBAL\_VARIABLE
- MASK
- METHOD
- MODULE
- PERMISSION
- PROCEDURE
- SPECIFIC
- TABLE
- TRIGGER
- TYPE
- USAGELIST
- VIEW

This value is not case-sensitive. This value can be a null value which means that all of the object types are considered.

If any of these types is specified, the procedure revalidates all of the invalid objects of that type, with the exception of those that belong to a MODULE. If you want to revalidate objects that are inside of a module, use the MODULE type with the name of a specific module, and all of the invalid objects inside of that module will be revalidated.

If there is a routine that has more than one parameter signature and you only want to revalidate one of them, use the SPECIFIC type with the name of the routine that you want to revalidate.

If you use the TABLE type, the specified tables will be reorganized and their statistics will be collected. The procedure invokes the reorg utility, followed by the runstats utility, against regular or materialized query tables that are in reorg-pending state. The procedure will attempt to use a user profile for runstats, if one exists. If not, a default runstats operation is invoked.

### *object\_schema*

An input argument of type VARCHAR(128) that identifies the schema name used to qualify database object references. The name is case-sensitive. This value can be NULL.

### *object\_name*

An input argument of type VARCHAR(128) that identifies a database object. The name is case-sensitive. This value cannot be the value of a typed table or a

row function, because the procedure does not support these types of objects; if the name of such an object is specified, an error is returned. This value can be NULL.

### ***force***

An input argument of type CHAR(1) to indicate if revalidation should be forced for the following objects (regardless of their current state).

- All SQL PL and PL/SQL objects if the invoking user holds DBADM authority.
- SQL PL and PL/SQL objects that the invoking user owns.

The argument is ignored for SQL PL and PL/SQL objects that do not meet the criteria listed previously and for any object that is not a SQL PL or PL/SQL object.

Values are:

- 'Y' or 'y' - All compiled SQL PL and PL/SQL objects will be revalidated.
- 'N', 'n' or NULL - Default behavior, only invalid objects will be revalidated.

### ***debug\_flag***

An input argument of type CHAR(1) to indicate whether debug mode is enabled or disabled for the following objects processed by the procedure:

- All SQL PL and PL/SQL objects if the invoking user holds DBADM authority or is a member of the built-in role, SYSDEBUG.
- SQL PL and PL/SQL objects that the invoking user owns.

The argument is ignored for SQL PL and PL/SQL objects that do not meet the criteria listed previously and for any object that is not a SQL PL or PL/SQL object.

Values are:

- 'Y' or 'y' - Compiled SQL PL and PL/SQL objects are revalidated with the debug mode enabled.
- 'N' or 'n' - Compiled SQL PL and PL/SQL objects are revalidated with the debug mode disabled.
- NULL - Compiled SQL PL and PL/SQL objects are revalidated with no change in the debug mode. This is the default behavior.

## **Authorization**

One of the following authorities is required:

- EXECUTE privilege on the ADMIN\_REVALIDATE\_DB\_OBJECTS procedure
- SECADM authority when object\_type is MASK or PERMISSION

When the value for the *force* option is 'Y', the invoking user must either be the owner of the object being revalidated or hold DBADM authority.

When the value for the *debug\_flag* option is 'Y', the invoking user must be the owner of the object, hold DBADM authority, or be a member of the SYSDEBUG system defined role.

## **Default PUBLIC privilege**

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

## Examples

*Example 1:* Revalidate everything in the current database.

```
CALL SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS(NULL, NULL, NULL)
```

Or, alternatively, call the procedure without any parameters.

```
CALL SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS()
```

*Example 2:* Revalidate all objects that are qualified by the schema MY\_SCHEMA.

```
CALL SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS(NULL, 'MY_SCHEMA', NULL)
```

*Example 3:* Revalidate all trigger objects in the database.

```
CALL SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS('trigger', NULL, NULL)
```

*Example 4:* Revalidate a specific view object.

```
CALL SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS('view', 'MY_SCHEMA', 'MY_VIEW')
```

*Example 5:* Revalidate all procedures under MY\_SCHEMA. In this example, there are three procedures (proc1, proc2, and proc3) under this schema. The referenced object used by proc1 does not exist. The following call revalidates proc2 and proc3, but proc1 remains invalid. In this situation, the call returns a warning.

```
CALL SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS('procedure', 'MY_SCHEMA', NULL)
```

*Example 6:* Revalidate an object that does not exist. This example returns an error.

```
CALL SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS('procedure', 'MY_SCHEMA', 'MY_VIEW')
```

*Example 7:* Revalidate all procedures under MY\_SCHEMA using the named parameter notation.

```
CALL SYSPROC.ADMIN_REVALIDATE_DB_OBJECTS(
 object_type=>'PROCEDURE', object_schema=>'MY_SCHEMA')
```

## Usage notes

All of the non-null parameter values that are passed to the ADMIN\_REVALIDATE\_DB\_OBJECTS procedure must be satisfied, or the procedure cannot identify the objects that need to be revalidated. For example, if you specify a view name, but with a trigger type, the procedure does not revalidate the view, because the type does not match.

This procedure will revalidate only invalid objects and regular or materialized query tables in reorg-pending state. All invalid objects can be found in SYSCAT.INVALIDOBJECTS. To find out which tables are in reorg-pending state, use the ADMIN\_GET\_TAB\_INFO table function.

If a valid object is specified as input, the procedure will not perform any operation and returns a success code. If a failure occurs during the revalidation of tables, the procedure fails. If a failure occurs during the revalidation of other objects, the procedure ignores the failure and continues revalidating the other objects. If there is at least one failure, the procedure returns a warning (SQLSTATE 0168B). If the revalidation of all objects fails, the procedure returns an error (SQLSTATE 429C4). The details of all revalidation failures of objects except tables can be found in SYSCAT.INVALIDOBJECTS.

In order to revalidate invalid masks or permissions, the user that runs ADMIN\_REVALIDATE\_DB\_OBJECTS must have SECADM authority. If there is at

least one failure, and the first failure is because the user does not have SECADM authority during revalidation of a mask or permission, the procedure returns a warning (SQLSTATE 0168B, SQLCODE +361), **msg-token2** contains CREATE PERMISSION or CREATE MASK. If the revalidation of all objects fails, and the first failure is because the user does not have SECADM authority during revalidation of a mask or permission, the procedure returns an error (SQLSTATE 42501, SQLCODE -551).

When a global variable is revalidated, it is also instantiated for the current session.

To monitor the progress of a table revalidation, you can monitor the progress of the associated table reorg operation. For all other objects, query the SYSCAT.INVALIDOBJECTS catalog view; objects are deleted from this view when they are successfully revalidated, and entries are updated if revalidation fails.

To ensure consistency of catalogs data, the ADMIN\_REVALIDATE\_DB\_OBJECTS routine is designed to run explicit COMMIT statements. Since the ADMIN\_REVALIDATE\_DB\_OBJECTS routine runs explicit COMMIT statements, it might not be possible to use a ADMIN\_REVALIDATE\_DB\_OBJECTS routine as a part of a Unit of Work (UOW). For example, when a UOW runs a ROLLBACK statement, any changes that are made to database objects previous to and by the ADMIN\_REVALIDATE\_DB\_OBJECTS routine before the ROLLBACK statement was run remains in effect.

### **ADMIN\_SET\_INTRA\_PARALLEL procedure - Enables or disables intrapartition parallelism**

The ADMIN\_SET\_INTRA\_PARALLEL procedure enables or disables intrapartition parallelism for a database application. Although the procedure is called in the current transaction, it takes effect starting with the next transaction.

#### **Syntax**

►►—ADMIN\_SET\_INTRA\_PARALLEL—(—*state*—)—————►►

The schema is SYSPROC.

#### **Procedure parameters**

##### ***state***

An input argument of type VARCHAR(3) that specifies the required state of intrapartition parallelism for the database application. The argument can be one of the following values:

##### ***YES, or yes***

The database application starts to run with intrapartition parallelism enabled starting with the next transaction.

##### ***NO, no***

The database application starts to run with intrapartition parallelism disabled starting with the next transaction.

##### ***NULL***

The database application starts to run with the intrapartition parallelism state dependent on the value of the **intra\_parallel** database manager configuration parameter, starting with the next transaction.

## Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

None

## Examples

*Example 1:* To run the database application with intrapartition parallelism enabled from the next transaction:

```
CALL ADMIN_SET_INTRA_PARALLEL('YES')
```

*Example 2:* To run the database application with intrapartition parallelism disabled from the next transaction:

```
CALL ADMIN_SET_INTRA_PARALLEL('NO')
```

*Example 3:* To run the database application with intrapartition parallelism state dependent on the value of the **intra\_parallel** database manager configuration parameter:

```
CALL ADMIN_SET_INTRA_PARALLEL(NULL)
```

## Usage notes

The required state of parallelism set through ADMIN\_SET\_INTRA\_PARALLEL only takes effect for the application invoking this procedure.

Calls to ADMIN\_SET\_INTRA\_PARALLEL will not change the **intra\_parallel** database manager configuration parameter setting.

Intrapartition parallelism settings applied by ADMIN\_SET\_INTRA\_PARALLEL can be overridden if the database application is associated with a workload that has specified a value for the MAX DEGREE workload attribute.

Intrapartition parallelism cannot be enabled by this procedure if the shared sort heap is not available (SQL5192W).

## ADMINTABINFO administrative view and ADMIN\_GET\_TAB\_INFO table function - retrieve table size and state information

The ADMINTABINFO administrative view and the ADMIN\_GET\_TAB\_INFO table function provide methods to retrieve table size and state information that is not currently available in the catalog views.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “ADMINTABINFO administrative view” on page 224
- “ADMIN\_GET\_TAB\_INFO table function” on page 225

## ADMINTABINFO administrative view

The ADMINTABINFO administrative view returns size and state information for tables, materialized query tables (MQT) and hierarchy tables only. These table types are reported as T for table, S for materialized query tables and H for hierarchy tables in the SYSCAT.TABLES catalog view. The information is returned at both the data partition level and the database partition level for a table.

In a Db2 pureScale environment, values reported for a table are identical on all members since all members operate on a single physical partition of data. This is unlike in a partitioned database environment, where each member operates on a different physical partition of data, and reports different values. Because the values on all members are always the same, the ADMINTABINFO view and the ADMIN\_GET\_TAB\_INFO table functions return only a single row for each table when run on a Db2 pureScale instance.

The schema is SYSIBMADM.

Refer to the Information returned by ADMINTABINFO administrative view and ADMIN\_GET\_TAB\_INFO table for a complete list of information that can be returned.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the ADMINTABINFO administrative view
- CONTROL privilege on the ADMINTABINFO administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

### Examples

*Example 1:* Retrieve size and state information for all tables

```
SELECT * FROM SYSIBMADM.ADMINTABINFO
```

*Example 2:* Determine the amount of physical space used by a large number of sparsely populated tables.

```
SELECT TABSCHEMA, TABNAME, SUM(DATA_OBJECT_P_SIZE),
 SUM(INDEX_OBJECT_P_SIZE), SUM(LONG_OBJECT_P_SIZE),
 SUM(LOB_OBJECT_P_SIZE), SUM(XML_OBJECT_P_SIZE)
FROM SYSIBMADM.ADMINTABINFO GROUP BY TABSCHEMA, TABNAME
```

*Example 3:* Identify tables that are eligible to use large RIDs, but are not currently enabled to use large RIDs.

```
SELECT TABSCHEMA, TABNAME FROM SYSIBMADM.ADMINTABINFO
WHERE LARGE_RIDS = 'P'
```

*Example 4:* Check the current type of statistics information collected for table T1

```
SELECT SUBSTR(TABSCHEMA, 1, 10) AS TABSCHEMA, SUBSTR(TABNAME, 1, 10)
 AS TBNAME, STATSTYPE FROM SYSIBMADM.ADMINTABINFO WHERE TABNAME = 'T1';
```

```
TABSCHEMA TBNAME STATSTYPE

DB2USER1 T1 U
```

1 record(s) selected.

## ADMIN\_GET\_TAB\_INFO table function

The ADMIN\_GET\_TAB\_INFO table function returns the same information as the ADMINTABINFO administrative view, but allows you to specify a schema and table name.

Refer to the Information returned by ADMINTABINFO administrative view and ADMIN\_GET\_TAB\_INFO table for a complete list of information that can be returned.

## Authorization

EXECUTE privilege on the ADMIN\_GET\_TAB\_INFO table function.

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Syntax

```
►►—ADMIN_GET_TAB_INFO—(—tabschema—,—tabname—)——————►►
```

The schema is SYSPROC.

## Routine parameters

*tabschema*

An input argument of type VARCHAR(128) that specifies a schema name.

*tabname*

An input argument of type VARCHAR(128) that specifies a table name, a materialized query table name or a hierarchy table name.

## Information returned

Table 46. Information Returned by ADMINTABINFO administrative view and ADMIN\_GET\_TAB\_INFO

Column name	Data type	Description
TABSCHEMA	VARCHAR(128)	table_schema - Table schema name monitor element
TABNAME	VARCHAR(128)	table_name - Table name monitor element
TABTYPE	CHAR(1)	Table type: <ul style="list-style-type: none"> <li>• 'H' = hierarchy table</li> <li>• 'S' = materialized query table</li> <li>• 'T' = table</li> </ul>
DBPARTITIONNUM	SMALLINT	dbpartitionnum - Database partition number monitor element

Table 46. Information Returned by ADMINTABINFO administrative view and ADMIN\_GET\_TAB\_INFO (continued)

Column name	Data type	Description
DATA_PARTITION_ID	INTEGER	data_partition_id - Data partition identifier monitor element
AVAILABLE	CHAR(1)	<p>State of the table:</p> <ul style="list-style-type: none"> <li>• 'N' = the table is unavailable. If the table is unavailable, all other output columns relating to the size and state will be NULL.</li> <li>• 'Y' = the table is available.</li> </ul> <p><b>Note:</b> Rollforward through an unrecoverable load will put a table into the unavailable state.</p>
DATA_OBJECT_L_SIZE	BIGINT	<p>Data object logical size. Amount of disk space logically allocated for the table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated for the table (for example, in the case of a logical table truncation). For multidimensional clustering (MDC) and insert time clustering (ITC) tables, this size includes the logical size of the block map object. The size returned takes into account full extents that are logically allocated for the table and, for objects created in DMS table spaces, an estimate of the Extent Map Page (EMP) extents. This size represents the logical size of the base table only. Space consumed by LOB data, Long Data, Indexes and XML objects are reported by other columns.</p>
DATA_OBJECT_P_SIZE	BIGINT	<p>Data object physical size, reported in kilobytes.</p> <p>For row-organized tables, the data object physical size is the amount of disk space that is physically allocated for the table. For MDC and ITC tables, this size includes the size of the block map object. The size that is returned takes into account full extents that are allocated for the table and includes the EMP extents for objects that you create in DMS table spaces. This size represents the physical size of the base table only. The space that is consumed by LOB data, long data, indexes, and XML objects is reported in other columns.</p> <p>For column-organized tables, the data object physical size is the amount of disk space that is physically allocated for the table metadata only, which is relatively small. The user data for column-organized tables is stored in the column-organized data object, and its physical size is reported by COL_OBJECT_P_SIZE.</p>

Table 46. Information Returned by ADMIN\_TABINFO administrative view and ADMIN\_GET\_TAB\_INFO (continued)

Column name	Data type	Description
INDEX_OBJECT_L_SIZE	BIGINT	<p>Index object logical size. Amount of disk space logically allocated for the indexes defined on the table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold index data for the table (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for the indexes and, for indexes created in DMS table spaces, an estimate of the EMP extents.</p> <p>For partitioned indexes on partitioned tables, this is the logical size of the index object containing index partitions for the data partition identified by DATA_PARTITION_ID. This value does not take into account nonpartitioned indexes on partitioned tables. For information about the both partitioned and nonpartitioned indexes, you can use the ADMIN_GET_INDEX_INFO function.</p>
INDEX_OBJECT_P_SIZE	BIGINT	<p>Index object physical size. Amount of disk space physically allocated for the indexes defined on the table, reported in kilobytes. The size returned takes into account full extents allocated for the indexes and includes the EMP extents for indexes created in DMS table spaces.</p> <p>For partitioned indexes on partitioned tables, this is the physical size of the index object containing index partitions for the data partition identified by DATA_PARTITION_ID. This value does not take into account nonpartitioned indexes on partitioned tables. For information about both partitioned and nonpartitioned indexes, you can use the ADMIN_GET_INDEX_INFO function.</p>
LONG_OBJECT_L_SIZE	BIGINT	<p>Long object logical size. Amount of disk space logically allocated for long field data in a table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold long field data for the table (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for long field data and, for long field data created in DMS table spaces, an estimate of the EMP extents.</p>
LONG_OBJECT_P_SIZE	BIGINT	<p>Long object physical size. Amount of disk space physically allocated for long field data in a table, reported in kilobytes. The size returned takes into account full extents allocated for long field data and includes the EMP extents for long field data created in DMS table spaces.</p>

Table 46. Information Returned by ADMIN\_TABINFO administrative view and ADMIN\_GET\_TAB\_INFO (continued)

Column name	Data type	Description
LOB_OBJECT_L_SIZE	BIGINT	LOB object logical size. Amount of disk space logically allocated for LOB data in a table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold LOB data for the table (for example, in the case of a logical table truncation). The size includes space logically allocated for the LOB allocation object. The size returned takes into account full extents that are logically allocated for LOB data and, for LOB data created in DMS table spaces, an estimate of the EMP extents.
LOB_OBJECT_P_SIZE	BIGINT	LOB object physical size. Amount of disk space physically allocated for LOB data in a table, reported in kilobytes. The size includes space allocated for the LOB allocation object. The size returned takes into account full extents allocated for LOB data and includes the EMP extents for LOB data created in DMS table spaces.
XML_OBJECT_L_SIZE	BIGINT	XML object logical size. Amount of disk space logically allocated for XML data in a table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold XML data for the table (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for XML data and, for XML data created in DMS table spaces, an estimate of the EMP extents.
XML_OBJECT_P_SIZE	BIGINT	XML object physical size. Amount of disk space physically allocated for XML data in a table, reported in kilobytes. The size returned takes into account full extents allocated for XML data and includes the EMP extents for XML data created in DMS table spaces.
INDEX_TYPE	SMALLINT	Indicates the type of indexes currently in use for the table. Returns 2 as type-2 indexes are used.
REORG_PENDING	CHAR(1)	A value of 'Y' indicates that a reorg recommended alter has been applied to the table and a classic (offline) reorg is required. Otherwise 'N' is returned.
INPLACE_REORG_STATUS	VARCHAR(10)	Current status of an inplace table reorganization on the table. The status value can be one of the following values: <ul style="list-style-type: none"> <li>• ABORTED (in a PAUSED state, but unable to RESUME; STOP is required)</li> <li>• EXECUTING</li> <li>• NULL (if no inplace reorg has been performed on the table)</li> <li>• PAUSED</li> </ul>
LOAD_STATUS	VARCHAR(12)	Current status of a load operation against the table. The status value can be one of the following values: <ul style="list-style-type: none"> <li>• IN_PROGRESS</li> <li>• NULL (if there is no load in progress for the table and the table is not in load pending state)</li> <li>• PENDING</li> </ul>

Table 46. Information Returned by ADMIN\_TABINFO administrative view and ADMIN\_GET\_TAB\_INFO (continued)

Column name	Data type	Description
READ_ACCESS_ONLY	CHAR(1)	'Y' if the table is in Read Access Only state, 'N' otherwise. A value of 'N' should not be interpreted as meaning that the table is fully accessible. If a load is in progress or pending, a value of 'Y' means the table data is available for read access, and a value of 'N' means the table is inaccessible. Similarly, if the table status is set integrity pending (refer to SYSCAT.TABLES STATUS column), then a value of 'N' means the table is inaccessible.
NO_LOAD_RESTART	CHAR(1)	A value of 'Y' indicates the table is in a partially loaded state that will not allow a load restart. A value of 'N' is returned otherwise.
NUM_REORG_REC_ALTERS	SMALLINT	Number of reorg recommend alter operations (for example, alter operations after which a reorganization is required) that have been performed against this table since the last reorganization.
INDEXES_REQUIRE_REBUILD	CHAR(1)	For nonpartitioned tables, 'Y' if any of the indexes defined on the table require a rebuild, and 'N' otherwise. For partitioned tables, 'Y' if any index partitions for the data partition identified by DATA_PARTITION_ID require a rebuild, and 'N' otherwise.
LARGE_RIDS	CHAR(1)	Indicator of whether the table is using large row IDs (RID, a 4-byte page number and a 2-byte slot number). A value of 'Y' indicates that the table is using large RIDs, and 'N' indicates that it is not using large RIDs. A value of 'P' (pending) is returned if both the following criteria are met: <ul style="list-style-type: none"> <li>• The table supports large RIDs, that is, the table is in a large table space.</li> <li>• At least one of the indexes for the table has not been reorganized or rebuilt yet, so the table is still using 4-byte RIDs. You must convert the table or indexes.</li> </ul> This column returns the null value for column-organized tables because the concept of RIDs does not apply.
LARGE_SLOTS	CHAR(1)	Indicator of whether the table is using large slots, which allows more than 255 rows per page. A value of 'Y' indicates that the table is using large slots, and 'N' indicates that it is not using large slots. A value of 'P' (pending) is returned if the following criteria are both met: <ul style="list-style-type: none"> <li>• The table supports large slots, that is, the table is in a large table space.</li> <li>• No offline table reorganization or table truncation operation has been performed on the table yet, so it is still using a maximum of 255 rows per page.</li> </ul> This column returns the null value for column-organized tables because the concept of slots does not apply.
DICTIONARY_SIZE	BIGINT	Size of the table dictionary, in bytes, used for row compression if a row compression dictionary exists for the table. If a historical dictionary exists, this value is the sum of the current and historical dictionary sizes.
BLOCKS_PENDING_CLEANUP	BIGINT	blocks_pending_cleanup - Pending cleanup rolled-out blocks monitor element

Table 46. Information Returned by ADMIN\_TABINFO administrative view and ADMIN\_GET\_TAB\_INFO (continued)

Column name	Data type	Description
STATSTYPE	CHAR(1)	<ul style="list-style-type: none"> <li>'F' = System fabricated statistics without table or index scan. These statistics are stored in memory and are different from what is stored in the system catalogs. This is a temporary state and eventually full statistics will be gathered by the database system and stored in the system catalogs.</li> <li>'A' = System asynchronously gathered statistics. Statistics have been automatically collected by the database system by a background process and stored in the system catalogs.</li> <li>'S' = System synchronously gathered statistics. Statistics have been automatically collected by the database system during SQL statement compilation. These statistics are stored in memory and are different from what is stored in the system catalogs. This is a temporary state and eventually the database system stores the statistics in the system catalogs.</li> <li>'U' = User gathered statistics. Statistics gathering was initiated by the user through a utility such as RUNSTATS, CREATE INDEX, LOAD, REDISTRIBUTE or by manually updating system catalog statistics.</li> <li>NULL = unknown type</li> </ul>
XML_RECORD_TYPE	SMALLINT	<p>Indicates the type of XML record currently in use for the table.</p> <ul style="list-style-type: none"> <li>1 if the type-1 (single node) XML record format is being used.</li> <li>2 if the type-2 (multi-node) XML record format is being used.</li> <li>Null if the table has no XML columns.</li> </ul>
RECLAIMABLE_SPACE	BIGINT	Indicates the amount of disk space that can be reclaimed by running the <b>REORG TABLE</b> command with the RECLAIM EXTENTS option for an MDC or ITC table in a DMS table space. For a column-organized table, this value indicates the amount of free space that was found the last time that the <b>RUNSTATS</b> command was run. Disk space is reported in kilobytes. For any other table, the value is zero.
XML_DICTIONARY_SIZE	BIGINT	Size of the XML dictionary, in bytes, used for data compression if a data compression dictionary exists for the XML storage object. If the table does not contain any XML columns or if a compression dictionary has not been created, the value is 0.
AMT_STATUS	VARCHAR(12)	Current status of ADMIN_MOVE_TABLE stored procedure call against the table. Returns the value 'IN_PROGRESS' or the null value if there is no move in progress for the table.
SPARSE_BLOCKS	BIGINT	For an insert time clustering (ITC) table, this field represents the number of sparsely used blocks in the table. This is the set of blocks that will be consolidated if the REORG TABLE command is invoked with the RECLAIM EXTENTS option.
STATS_ROWS_MODIFIED	BIGINT	stats_rows_modified - Rows modified since last RUNSTATS

Table 46. Information Returned by ADMIN\_TABINFO administrative view and ADMIN\_GET\_TAB\_INFO (continued)

Column name	Data type	Description
RTS_ROWS_MODIFIED	BIGINT	rts_rows_modified - Rows modified since last real time statistics
STATS_DBPARTITION	CHAR(3)	stats_dbpartition - Automatics statistics collection indicator
COL_OBJECT_L_SIZE	BIGINT	col_object_l_size - Column-organized data object logical size
COL_OBJECT_P_SIZE	BIGINT	col_object_p_size - Column-organized data object physical size

### Usage notes

- If both the *tabschema* and *tabname* are specified, information is returned for that specific table only.
- If the *tabschema* is specified but *tabname* is NULL or the empty string ("), then information is returned for all tables in the given schema.
- If the *tabschema* is NULL or the empty string (") and *tabname* is specified, then an error is returned. To retrieve information for a specific table, the table must be identified by both schema and table name.
- If both *tabschema* and *tabname* are NULL or the empty string ("), then information is returned for all tables.
- If *tabschema* or *tabname* do not exist, or *tabname* does not correspond to a table name (type T), a materialized query table name (type S) or a hierarchy table name (type H), an empty result set is returned.
- When the ADMIN\_GET\_TAB\_INFO table function is retrieving data for a given table, it will acquire a shared lock on the corresponding row of SYSTABLES to ensure consistency of the data that is returned (for example, to ensure that the table is not dropped while information is being retrieved for it). The lock will only be held for as long as it takes to retrieve the size and state information for the table, not for the duration of the table function call.
- Physical size reported for tables in SMS table spaces is the same as logical size.
- When an inplace reorg is active on a table, the physical size for the data object (DATA\_OBJECT\_P\_SIZE) will not be calculated. Only the logical size will be returned. You can tell if an inplace reorg is active on the table by looking at the INPLACE\_REORG\_STATUS output column.

### Examples

*Example 1:* Retrieve size and state information for the table DBUSER1.EMPLOYEE.

```
SELECT * FROM TABLE (SYSPROC.ADMIN_GET_TAB_INFO('DBUSER1', 'EMPLOYEE'))
 AS T
```

*Example 2:* Suppose there exists a non-partitioned table (DBUSER1.EMPLOYEE), with all associated objects (for example, indexes and LOBs) stored in a single table space. Calculate how much physical space the table is using in the table space:

```
SELECT (data_object_p_size + index_object_p_size + long_object_p_size +
 lob_object_p_size + xml_object_p_size + col_object_p_size) as total_p_size
FROM TABLE(SYSPROC.ADMIN_GET_TAB_INFO('DBUSER1', 'EMPLOYEE')) AS T
```

Calculate how much space would be required if the table were moved to another table space, where the new table space has the same page size and extent size as the original table space:

```
SELECT (data_object_l_size + index_object_l_size + long_object_l_size +
lob_object_l_size + xml_object_l_size + col_object_l_size) as total_l_size
FROM TABLE(SYSPROC.ADMIN_GET_TAB_INFO('DBUSER1', 'EMPLOYEE')) AS T
```

*Example 3:* Determine the total size for the compression dictionaries for the table DBUSER1.EMPLOYEE.

```
SELECT SUBSTR(TABSCHEMA,1,10) AS TBSHEMA, SUBSTR(TABNAME,1,10) AS TBNAME,
DICTIONARY_SIZE + XML_DICTIONARY_SIZE AS TOTAL_DICTIONARY_SIZE
FROM TABLE(SYSPROC.ADMIN_GET_TAB_INFO('DBUSER1','EMPLOYEE'))
```

*Example 4:* Determine the amount of space reclaimable from a multidimensional clustering table SAMPLE.STAFF:

```
SELECT RECLAIMABLE_SPACE
FROM TABLE(SYSPROC.ADMIN_GET_TAB_INFO('SAMPLE','STAFF'))
```

## **ADMINTEMPCOLUMNS administrative view and ADMIN\_GET\_TEMP\_COLUMNS table function - Retrieve column information for temporary tables**

The ADMINTEMPCOLUMNS administrative view and the ADMIN\_GET\_TEMP\_COLUMNS table function provide methods to retrieve column attribute information for created temporary tables and declared temporary tables.

Although the catalog views contain column attribute information for instances of created temporary tables, they do not have this information for declared temporary tables.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “ADMINTEMPCOLUMNS administrative view” on page 233
- “ADMIN\_GET\_TEMP\_COLUMNS table function” on page 234

### **ADMINTEMPCOLUMNS administrative view**

The ADMINTEMPCOLUMNS administrative view returns column attribute information for instances of created temporary tables and declared temporary tables.

The schema is SYSIBMADM.

Refer to the Information returned for ADMINTEMPCOLUMNS administrative view and ADMIN\_GET\_TEMP\_COLUMNS table for a complete list of information that can be returned.

### **Authorization**

One of the following authorizations is required:

- SELECT privilege on the ADMINTEMPCOLUMNS administrative view
- CONTROL privilege on the ADMINTEMPCOLUMNS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the ADMIN\_GET\_TEMP\_COLUMNS table function

- DATAACCESS authority

## Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

## Examples

*Example 1:* Retrieve column attribute information for all instances of created temporary tables and all declared temporary tables currently existing in the database.

```
SELECT * FROM SYSIBMADM.ADMINTEMPCOLUMNS
```

*Example 2:* Determine which temporary tables active in the database are using the user-defined data type of USMONEY.

```
SELECT APPLICATION_HANDLE, TABSCHEMA, TABNAME
FROM SYSIBMADM.ADMINTEMPCOLUMNS
WHERE TYPENAME = 'USMONEY'
```

*Example 3:* Retrieve table schema, table name, and the column names for all declared temporary tables declared by the SYSTEM\_USER.

```
SELECT T.TABSCHEMA, T.TABNAME, C.COLNAME
FROM SYSIBMADM.ADMINTEMPCOLUMNS C, SYSIBMADM.ADMINTEMPTABLES T
WHERE T.TEMPTABTYPE = 'D'
AND T.INSTANTIATOR = SYSTEM_USER
AND T.TABSCHEMA = C.TABSCHEMA
AND T.TABNAME = C.TABNAME
```

## ADMIN\_GET\_TEMP\_COLUMNS table function

The ADMIN\_GET\_TEMP\_TABLES table function returns the same information as the ADMINTEMPCOLUMNS administrative view, but allows you to specify a schema name and a table name.

Refer to the Information returned for ADMINTEMPCOLUMNS administrative view and ADMIN\_GET\_TEMP\_COLUMNS table for a complete list of information that can be returned.

## Syntax

```
►►—ADMIN_GET_TEMP_COLUMNS—(—application_handle—,—tabschema—,—tabname—)—►◄
```

The schema is SYSPROC.

## Table function parameters

### *application\_handle*

An input argument of type BIGINT that specifies an application handle. If *application\_handle* is specified, data is returned for the specified connection only; if *application\_handle* is NULL, data is returned for all connections.

### *tabschema*

An input argument of type VARCHAR(128) that specifies a schema name.

*tabname*

An input argument of type VARCHAR(128) that specifies a created temporary table name or a declared temporary table name.

## Authorization

One of the following authorizations is required:

- EXECUTE privilege on the ADMIN\_GET\_TEMP\_COLUMNS table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Examples

*Example 1:* Retrieve column information for the declared temporary table TEMPEMPOLOYEE for the current connection.

```
SELECT *
 FROM TABLE (
 SYSPROC.ADMIN_GET_TEMP_COLUMNS(
 APPLICATION_ID(), 'SESSION', 'TEMPEMPOLOYEE'))
AS T
```

## Usage notes

- If both *tabschema* and *tabname* are specified, then information is returned for that specific temporary table only.
- If *tabschema* is specified but *tabname* is NULL or the empty string ("), then information is returned for all tables in the given schema.
- If *tabschema* is NULL or the empty string (") and *tabname* is specified, then an error is returned. To retrieve information for a specific temporary table, the table must be identified by both schema and table name.
- If both *tabschema* and *tabname* are NULL or the empty string ("), then information is returned for all temporary tables for the connection or all connections, depending on the value of *application\_handle*.
- If *tabschema* or *tabname* do not exist, or *tabname* does not correspond to a temporary table name, or instances of the identified temporary table do not exist in the database, then an empty result set is returned.

## Information returned

Table 47. Information returned for ADMINTEMPCOLUMNS administrative view and ADMIN\_GET\_TEMP\_COLUMNS

Column name	Data type	Description
APPLICATION_HANDLE	BIGINT	application_handle - Application handle monitor element
APPLICATION_NAME	VARCHAR(256)	appl_name - Application name monitor element
TABSCHEMA	VARCHAR(128)	table_schema - Table schema name monitor element
TABNAME	VARCHAR(128)	table_name - Table name monitor element
COLNAME	VARCHAR(128)	Name of the column.
COLNO	SMALLINT	Number of this column in the table (starting with 0).

Table 47. Information returned for ADMIN\_TEMP\_COLUMNS administrative view and ADMIN\_GET\_TEMP\_COLUMNS (continued)

Column name	Data type	Description
TYPE_SCHEMA	VARCHAR(128)	Schema name of the data type for the column.
TYPE_NAME	VARCHAR(128)	Unqualified name of the data type for the column.
LENGTH	INTEGER	Maximum length of the data. 0 for distinct types. The LENGTH column indicates precision for DECIMAL fields, and indicates the number of bytes of storage required for decimal floating-point columns; that is, 8 columns for DECFLOAT(16) and 16 columns for DECFLOAT(34).
SCALE	SMALLINT	Scale if the column type is DECIMAL; or the number of digits of fractional seconds if the column type is TIMESTAMP; 0 otherwise.
DEFAULT	VARCHAR(254)	Default value for the column of a table expressed as a constant, special register, or cast-function appropriate for the data type of the column. Can also be the keyword NULL. Values might be converted from what was specified as a default value. For example, date and time constants are shown in ISO format, cast-function names are qualified with schema names, and identifiers are delimited. Null value if a DEFAULT clause was not specified or the column is a view column.
NULLS	CHAR(1)	Nullability attribute for the column. <ul style="list-style-type: none"> <li>• "Y" = Column is nullable</li> <li>• "N" = Column is not nullable</li> </ul> The value can be "N" for a view column that is derived from an expression or function. Nevertheless, such a column allows null values when the statement using the view is processed with warnings for arithmetic errors.
CODEPAGE	SMALLINT	Code page used for data in this column; 0 if the column is defined as FOR BIT DATA or is not a string type.
LOGGED	CHAR(1)	Applies only to columns whose type is LOB or distinct based on LOB; blank otherwise. <ul style="list-style-type: none"> <li>• "Y" = Column is logged</li> <li>• "N" = Column is not logged</li> </ul>
COMPACT	CHAR(1)	Applies only to columns whose type is LOB or distinct based on LOB; blank otherwise. <ul style="list-style-type: none"> <li>• "Y" = Column is compacted in storage</li> <li>• "N" = Column is not compacted</li> </ul>
INLINE_LENGTH	INTEGER	Maximum size in bytes of the internal representation of an instance of an XML document or a structured type that can be stored in the base table; 0 when not applicable.
IDENTITY	CHAR(1)	<ul style="list-style-type: none"> <li>• "Y" = Identity column</li> <li>• "N" = Not an identity column</li> </ul>
GENERATED	CHAR(1)	Type of generated column. <ul style="list-style-type: none"> <li>• "A" = Column value is always generated</li> <li>• "D" = Column values is generated by default</li> <li>• Blank = Column is not generated</li> </ul>

## **ADMINTEMPTABLES administrative view and ADMIN\_GET\_TEMP\_TABLES table function - Retrieve information for temporary tables**

The ADMINTEMPTABLES administrative view and the ADMIN\_GET\_TEMP\_TABLES table function provide methods to retrieve table attribute and instantiation time information for instances of created temporary tables and declared temporary tables.

Although the catalog views contain table attribute information for created temporary tables, they do not contain this information for declared temporary tables. In addition, the catalog views do not contain table instantiation time information for created temporary tables or declared temporary tables.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “ADMINTEMPTABLES administrative view” on page 237
- “ADMIN\_GET\_TEMP\_TABLES table function” on page 238

### **ADMINTEMPTABLES administrative view**

The ADMINTEMPTABLES administrative view returns table attribute and instantiation time information for instances of created temporary tables and declared temporary tables.

The schema is SYSIBMADM.

Refer to the Information returned for ADMINTEMPTABLES administrative view and ADMIN\_GET\_TEMP\_TABLES table for a complete list of information that can be returned.

### **Authorization**

One of the following authorizations is required:

- SELECT privilege on the ADMINTEMPTABLES administrative view
- CONTROL privilege on the ADMINTEMPTABLES administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the ADMIN\_GET\_TEMP\_TABLES table function
- DATAACCESS authority

### **Default PUBLIC privilege**

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

### **Examples**

*Example 1:* Retrieve table attributes and instantiation time information for all instances of created temporary tables and declared temporary tables currently existing in the database.

```
SELECT * FROM SYSIBMADM.ADMINTEMPTABLES
```

*Example 2:* Determine which connections have an instance of a created temporary table.

```
SELECT APPLICATION_HANDLE, TABSCHEMA, TABNAME
FROM SYSIBMADM.ADMINTEMPTABLES
WHERE TEMPTABTYPE = 'C'
```

*Example 3:* Retrieve table attributes and instantiation time information for all declared temporary tables declared for all the tables instantiated by the user that connected to the database.

```
SELECT TABSCHEMA, TABNAME, ONCOMMIT, ONROLLBACK,
INSTANTIATION_TIME
FROM SYSIBMADM.ADMINTEMPTABLES
WHERE TEMPTABTYPE = 'D' AND INSTANTIATOR = SYSTEM_USER
```

## ADMIN\_GET\_TEMP\_TABLES table function

The ADMIN\_GET\_TEMP\_TABLES table function returns the same information as the ADMINTABINFO administrative view, but allows you to specify a schema name and a table name.

Refer to the Information returned for ADMINTEMPTABLES administrative view and ADMIN\_GET\_TEMP\_TABLES table for a complete list of information that can be returned.

## Syntax

►►—ADMIN\_GET\_TEMP\_TABLES—(—*application\_handle*—,—*tabschema*—,—*tabname*—)————►◄

The schema is SYSPROC.

## Table function parameters

### *application\_handle*

An input argument of type BIGINT that specifies an application handle. If *application\_handle* is specified, data is returned for the specified connection only; if *application\_handle* is NULL, data is returned for all connections.

### *tabschema*

An input argument of type VARCHAR(128) that specifies a schema name.

### *tabname*

An input argument of type VARCHAR(128) that specifies a created temporary table name or a declared temporary table name.

## Authorization

One of the following authorizations is required:

- EXECUTE privilege on the ADMIN\_GET\_TEMP\_TABLES table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Examples

*Example 1:* Retrieve table attributes and instantiation time information for all instances of the created temporary table DBUSER1.EMPLOYEE for all connections.

```
SELECT TABSCHEMA, TABNAME, ONCOMMIT, ONROLLBACK, INSTANTIATION_TIME
FROM TABLE (SYSPROC.ADMIN_GET_TEMP_TABLES(NULL, 'DBUSER1', 'EMPLOYEE'))
AS T
```

*Example 2:* Retrieve the instantiation time and table space ID for all instances of user temporary tables for the current connection.

```
SELECT TABSCHEMA, TABNAME, INSTANTIATION_TIME, TBSP_ID
FROM TABLE (SYSPROC.ADMIN_GET_TEMP_TABLES(APPLICATION_ID(), '', ''))
AS T
```

## Usage notes

- 
- If both *tabschema* and *tabname* are specified, then information is returned for that specific temporary table only.
- If *tabschema* is specified but *tabname* is NULL or the empty string ("), then information is returned for all tables in the given schema.
- If *tabschema* is NULL or the empty string (") and *tabname* is specified, then an error is returned. To retrieve information for a specific temporary table, the table must be identified by both schema and table name.
- If both *tabschema* and *tabname* are NULL or the empty string ("), then information is returned for all temporary tables for the connection or all connections, depending on the value of *application\_handle*.
- If *tabschema* or *tabname* do not exist, or *tabname* does not correspond to a temporary table name, or instances of the identified temporary table do not exist in the database, then an empty result set is returned.

## Information returned

Table 48. Information returned for ADMINTEMPTABLES administrative view and ADMIN\_GET\_TEMP\_TABLES

Column name	Data type	Description
APPLICATION_HANDLE	BIGINT	application_handle - Application handle monitor element
APPLICATION_NAME	VARCHAR(256)	appl_name - Application name monitor element
TABSCHEMA	VARCHAR(128)	table_schema - Table schema name monitor element
TABNAME	VARCHAR(128)	table_name - Table name monitor element
INSTANTIATOR	VARCHAR(128)	Authorization ID under which the created temporary table was instantiated or declared temporary table was declared.
INSTANTIATOR_TYPE	CHAR(1)	<ul style="list-style-type: none"><li>• "U" = The instantiator is an individual user</li></ul>
TEMPTABTYPE	CHAR(1)	Temporary table type: <ul style="list-style-type: none"><li>• "C" = Created temporary table</li><li>• "D" = Declared temporary table</li></ul>

Table 48. Information returned for ADMINEMPTYTABLES administrative view and ADMIN\_GET\_TEMP\_TABLES (continued)

Column name	Data type	Description
INSTANTIATION_TIME	TIMESTAMP	Time at which the created temporary table instance was instantiated or the declared temporary table was declared.
COLCOUNT	SMALLINT	Number of columns, including inherited columns (if any).
TAB_FILE_ID	BIGINT	table_file_id - The file ID (FID) for the table.
TBSP_ID	BIGINT	tablespace_id - An integer that uniquely represents a table space used by the current database.
PMAP_ID	SMALLINT	Identifier for the distribution map that is currently in use by this table.
PARTITION_MODE	CHAR(1)	Indicates how data is distributed among database partitions in a partitioned database system. <ul style="list-style-type: none"> <li>• "H" = Hashing</li> <li>• Blank = No database partitioning</li> </ul>
CODEPAGE	SMALLINT	Code page of the object. This is the default code page used for all character columns and expression-generated columns.
ONCOMMIT	CHAR(1)	Specifies the action taken on this table when a COMMIT operation is performed. <ul style="list-style-type: none"> <li>• "D" = Delete rows</li> <li>• "P" = Preserve rows</li> </ul>
ONROLLBACK	CHAR(1)	Specifies the action taken on this table when a ROLLBACK operation is performed. <ul style="list-style-type: none"> <li>• "D" = Delete rows</li> <li>• "P" = Preserve rows</li> </ul>
LOGGED	CHAR(1)	Specifies whether this table is logged. <ul style="list-style-type: none"> <li>• "N" = Not logged</li> <li>• "Y" = Logged</li> </ul>
TAB_ORGANIZATION	CHAR(1)	tab_organization - Data organization in table monitor element

## Administrative Task Scheduler routines and views

This grouping of routines and views can be used to manage scheduled tasks.

### ADMIN\_TASK\_ADD procedure - Schedule a new task

The ADMIN\_TASK\_ADD procedure schedules an administrative task, which is any piece of work that can be encapsulated inside a procedure.

#### Syntax

```

▶▶ ADMIN_TASK_ADD (—name—, —begin_timestamp—, —end_timestamp—, —————▶
▶max_invocations—, —schedule—, —procedure_schema—, —procedure_name—, —————▶
▶procedure_input—, —options—, —remarks—) —————▶▶

```

The schema is SYSPROC.

## Procedure parameters

### ***name***

An input argument of type VARCHAR(128) that specifies the name of the task. This argument cannot be NULL.

### ***begin\_timestamp***

An input argument of type TIMESTAMP that specifies the earliest time a task can begin execution. The value of this argument cannot be in the past, and it cannot be later than *end\_timestamp*.

When task execution begins depends on how this argument and the *schedule* argument are defined:

- If the *begin\_timestamp* argument is not NULL:
  - If the *schedule* argument is NULL, the task execution begins at *begin\_timestamp*.
  - If the *schedule* argument is not NULL, the task execution begins at the next scheduled time at or after *begin\_timestamp*.
- If the *begin\_timestamp* argument is NULL:
  - If the *schedule* argument is NULL, the task execution begins immediately.
  - If the *schedule* argument is not NULL, the task execution begins at the next scheduled time.

### ***end\_timestamp***

An input argument of type TIMESTAMP that specifies the last time that a task can begin execution. The value of this argument cannot be in the past, and it cannot be earlier than *begin\_timestamp*. If the argument is NULL, the task can continue to execute as scheduled indefinitely.

An executing task will not be interrupted at its *end\_timestamp*.

### ***max\_invocations***

An input argument of type INTEGER that specifies the maximum number of executions allowed for the task. If the argument is NULL, there is no limit to the number of times the task can execute. If the argument is 0, the task will not execute.

This value applies to the schedule if *schedule* is not NULL.

If both *end\_timestamp* and *max\_invocations* are specified, *end\_timestamp* takes precedence. That is, if the *end\_timestamp* timestamp is reached, even though the number of task executions so far has not reached the value of *max\_invocations*, the task will not be executed again.

### ***schedule***

An input argument of type VARCHAR(1024) that specifies a task execution schedule at fixed points in time. If the argument is NULL, the task is not scheduled at fixed points in time.

The *schedule* string must be specified using the UNIX cron format.

Multiple schedules are not supported.

### ***procedure\_schema***

An input argument of type VARCHAR(128) that specifies the schema of the procedure that this task will execute. This argument cannot be NULL.

### ***procedure\_name***

An input argument of type VARCHAR(128) that specifies the name of the procedure that this task will execute. This argument cannot be NULL.

***procedure\_input***

An input argument of type CLOB(2M) that specifies the input arguments of the procedure that this task will execute. This argument must contain an SQL statement that returns one row of data. The returned values will be passed as arguments to the procedure. If this argument is NULL, no arguments are passed to the procedure.

The number of columns returned by the SQL statement must match the total number (and type) of arguments for the procedure and must contain a single row. For output arguments, the value itself is ignored, but should be of the same SQL data type as the procedure requires.

This SQL statement is executed every time the task is executed. If the SQL statement fails, the task's status will be set to N0TRUN and specific SQLCODE information will be recorded. If the statement does not return a result set, does not return a row, returns multiple rows or result sets the task will not be executed. The task's status will be set to N0TRUN and SQLCODE SQL1465N will be set to indicate that this argument is invalid.

If the statement result contains serialized XML parameters, the total size of all XML parameters combined is limited to 256 kilobytes. If the result exceeds this threshold, the task's status will be set to N0TRUN. SQLCODE -302 and SQLSTATE 22001 will be set to indicate that data truncation has occurred.

To view the task's status, use the SYSTOOL.ADMIN\_TASK\_STATUS view

***options***

An input argument of type VARCHAR(512). This argument must be NULL.

***remarks***

An input argument of type VARCHAR(254) that specifies a description of the task. This argument is optional and can be NULL.

**Authorization**

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

**Default PUBLIC privilege**

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

**Usage notes**

The SYSTOOLSPACE table space must exist before you call the ADMIN\_TASK\_ADD procedure. If it does not exist, the procedure will return an SQL0204N error message.

When a task is scheduled, the authorization ID of the current session user is recorded. The scheduler switches to this session authorization ID when the executing the task.

The administrative task scheduler does not support the execution of procedures that perform a database connection without a specified user ID and password. For

example, the ADMIN\_CMD procedure can be used to perform a LOAD from a database. A connection to the source database is established using the user ID and password provided for the currently connected database. This type of LOAD operation cannot be executed by the task scheduler.

If invalid arguments are passed into the procedure, SQL0171N will be returned. The tokens of the message will indicate which argument is invalid and the name of the procedure.

The task cannot be scheduled for execution until the unit of work is committed and the scheduler has fetched the task definition.

The scheduler checks for new or updated tasks every 5 minutes. To ensure the task executes as expected, the earliest begin time, as defined by the *begin\_timestamp*, *end\_timestamp* and *schedule* arguments, should be at least 5 minutes after the unit of work commits.

The database must be active on all database partitions to ensure the task can be executed by the scheduler.

In a partitioned database environment, the ADMIN\_TASK\_ADD procedure can be called from any database partition. The scheduler, however, will execute all tasks from the catalog database partition.

The *begin\_timestamp*, *end\_timestamp*, and *schedule* are based on the server's time zone. Special attention is required when scheduling a task during the transition period of daylight savings time (DST). If the task is scheduled to run 2:01 AM and it is the time of year when the time springs forward, the task will not run as the clock skips from 2:00 AM to 3:00 AM. If the DST change occurs when the database is activated and there is no task scheduled during the DST change time, the first scheduled task after the DST change will not run. Tasks will run normally from the second task onwards. For example, if no task is scheduled between 2:00 AM to 3:00 AM but there is a task at 3:10 AM which is also the first task after DST change, this task will not run. This is independent of the time after the DST change in which the task is scheduled. For example, whether the first task after DST is scheduled one day or one month after the DST change, the task will not run as long as it is the first task after DST change and the database is kept active the entire time. On the other hand, when the time falls back an hour, tasks that were originally scheduled between 2:00 AM and 3:00 AM will execute twice. The user is responsible for making adjustments for daylight savings time to ensure their required behavior.

The scheduler will always commit after calling the procedure specified by *procedure\_schema* and *procedure\_name*. If a transaction roll back is required, the rollback must occur inside the procedure.

If the task name is not unique, the procedure will fail with SQL0601N.

## Example

*Example 1:* Create a task that performs an online TSM backup daily at 12:00 AM, with immediate effect:

```
CALL SYSPROC.ADMIN_TASK_ADD
('DAILY TSM BACKUP',
 CURRENT_TIMESTAMP,
 NULL,
```

```

NULL,
'0 0 * * *',
'SYSPROC',
'ADMIN_CMD',
'VALUES(''BACKUP DATABASE SALES ONLINE USE TSM WITHOUT PROMPTING'')',
NULL,
NULL)

```

*Example 2:* Schedule a task to flush an event monitor every hour:

1. Create an SQL procedure, in the PROD schema, that flushes an event monitor called "em":

```

CREATE PROCEDURE FLUSH_EVENT_MONITOR()
SPECIFIC FLUSH_EVENT_MONITOR
LANGUAGE SQL
BEGIN
DECLARE stmt VARCHAR(100) ;
SET stmt = 'FLUSH EVENT MONITOR em' ;
EXECUTE IMMEDIATE stmt ;
END

```

**Note:** The FLUSH EVENT MONITOR SQL statement cannot be called directly in the procedure. However, EXECUTE IMMEDIATE can be used.

2. Call ADMIN\_TASK\_ADD to schedule the task:

```

CALL SYSPROC.ADMIN_TASK_ADD
('FLUSH EVENT MONITOR EVERY HOUR',
NULL,
NULL,
NULL,
'0 0-23 * * *',
'PROD',
'FLUSH_EVENT_MONITOR',
NULL,
NULL,
NULL)

```

### UNIX cron format:

The UNIX cron format is used to specify time in the *schedule* parameter of the ADMIN\_TASK\_ADD and ADMIN\_TASK\_UPDATE procedures.

The cron format has five time and date fields separated by at least one blank. There can be no blank within a field value. Scheduled tasks are executed when the *minute*, *hour*, and *month of year* fields match the current time and date, and at least one of the two day fields (*day of month*, or *day of week*) match the current date.

Table 1 lists the time and date fields and their allowed values in cron format.

*Table 49. Field names and values for the UNIX cron format*

Field name	Allowed values
<i>minute</i>	0-59
<i>hour</i>	0-23
<i>day of month</i>	1-31

Table 49. Field names and values for the UNIX cron format (continued)

Field name	Allowed values
<i>month</i>	<ul style="list-style-type: none"> <li>• 1-12, where 1 is January, 2 is February, and so on.</li> <li>• Uppercase, lowercase and mixed-case three character strings, based on the English name of the month. For example: jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, or dec.</li> </ul>
<i>day of week</i>	<ul style="list-style-type: none"> <li>• 0-7, where 0 or 7 is Sunday, 1 is Monday, and so on.</li> <li>• Uppercase, and lowercase or mixed-case three character strings, based on the English name of the day: mon, tue, wed, thu, fri, sat, or sun.</li> </ul>

### Ranges and lists

Ranges of numbers are allowed. Ranges are two numbers separated with a hyphen. The specified range is inclusive. For example, the range 8-11 for an hour entry specifies execution at hours 8, 9, 10 and 11.

Lists are allowed. A list is a set of numbers or ranges separated by commas. For example:

1,2,5,9

0-4,8-12

### Unrestricted range

A field can contain an asterisk (\*), which represents all possible values in the field.

The day of a command's execution can be specified by two fields: *day of month* and *day of week*. If both fields are restricted by the use of a value other than the asterisk, the command will run when either field matches the current time. For example, the value 30 4 1,15 \* 5 causes a command to run at 4:30 AM on the 1st and 15th of each month, plus every Friday.

### Step values

Step values can be used in conjunction with ranges. The syntax *range/step* defines the range and an execution interval.

If you specify *first-last/step*, execution takes place at *first*, then at all successive values that are distant from *first* by *step*, until *last*.

For example, to specify command execution every other hour, use 0-23/2. This expression is equivalent to the value 0,2,4,6,8,10,12,14,16,18,20,22.

If you specify *\*/step*, execution takes place at every interval of *step* through the unrestricted range. For example, as an alternative to 0-23/2 for execution every other hour, use \*/2.

## Example

Table 2 lists values that you can use for the *schedule* argument in ADMIN\_TASK\_ADD or ADMIN\_TASK\_UPDATE procedures for various scheduling scenarios.

Table 50. Example task schedules and the appropriate schedule argument values

Desired task schedule	schedule value
2:10 PM every Monday	10 14 * * 1
Every day at midnight	0 0 * * *
Every weekday at midnight	0 0 * * 1-5
Midnight on 1st and 15th day of the month	0 0 1,15 * *
6:32 PM on the 17th, 21st and 29th of November plus each Monday and Wednesday in November each year	32 18 17,21,29 11 mon,wed

## ADMIN\_TASK\_LIST administrative view - Retrieve information about tasks in the scheduler

The ADMIN\_TASK\_LIST administrative view retrieves information about each task defined in the administrative task scheduler.

The schema is SYSTOOLS.

This view is created the first time the ADMIN\_TASK\_ADD procedure is called.

## Authorization

SELECT or CONTROL privilege on the ADMIN\_TASK\_LIST administrative view. Unless the database was created with the **RESTRICTIVE** option, SELECT privilege is granted to PUBLIC by default.

When you query the ADMIN\_TASK\_LIST view, it will only return the tasks that were created using your session authorization ID. If you have SYSADM, SYSCTRL, SYSMAINT, or DBADM authority, all tasks are returned.

## Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

## Example

Request the list of tasks in the scheduler:

```
SELECT * from SYSTOOLS.ADMIN_TASK_LIST
```

## Information returned

Table 51. Information returned by the ADMIN\_TASK\_LIST administrative view

Column name	Data type	Description
NAME	VARCHAR(128)	The name of the task.
TASKID	INTEGER	The task identifier.

Table 51. Information returned by the ADMIN\_TASK\_LIST administrative view (continued)

Column name	Data type	Description
OWNER	VARCHAR(128)	The session authorization ID of the user that created the task.
OWNERTYPE	VARCHAR(1)	The authorization ID type. Valid values are: • U - User
BEGIN_TIME	TIMESTAMP	The timestamp of when the task is first able to run. <sup>1</sup>
END_TIME	TIMESTAMP	The timestamp of when the task is last able to run. <sup>1</sup>  If this column is NULL, the task can run indefinitely unless MAX_INVOCATIONS is specified.
MAX_INVOCATIONS	INTEGER	The maximum number of executions allowed for the task. If this column is NULL, the task can run indefinitely unless END_TIME is specified.
SCHEDULE	VARCHAR(1024)	The schedule for the task, in UNIX cron format.
PROCEDURE_SCHEMA	VARCHAR(128)	The schema of the procedure that this task will execute.
PROCEDURE_NAME	VARCHAR(128)	The name of the procedure that this task will execute.
PROCEDURE_INPUT	CLOB(2M)	The input parameters of the procedure that this task will execute. If this column is NULL, there are no input parameters.
OPTIONS	VARCHAR(512)	Options that affect the behavior of the task.
UPDATE_TIME	TIMESTAMP	The timestamp when the task was last updated.
REMARKS	VARCHAR(254)	A description of the task.

**Note:**

- <sup>1</sup> The BEGIN\_TIME and END\_TIME are based on the database server's time zone. The user is responsible for making adjustments for daylight savings time (DST).

**ADMIN\_TASK\_REMOVE procedure - Remove scheduled tasks or task status records**

The ADMIN\_TASK\_REMOVE procedure removes scheduled administrative tasks, which are pieces of work that can be encapsulated inside a procedure. It also removes task status records.

**Syntax**

►► ADMIN\_TASK\_REMOVE (—name—, —end\_timestamp—) ◀◀

The schema is SYSPROC.

## Procedure parameters

### *name*

An input argument of type VARCHAR(128) that specifies the name of the task.

### *end\_timestamp*

An output argument of type TIMESTAMP that specifies the status record *end\_timestamp* timestamp.

## Authorization

One of the following authorizations is required:

- EXECUTE privilege on the ADMIN\_TASK\_REMOVE procedure
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Although the statement authorization ID might allow the procedure to be executed, successful removal of task and status records depends on the value of the current session authorization ID. The current session authorization ID must match the session authorization ID that was recorded when the task was created. Users with SYSADM, SYSCTRL, SYSMAINT, or DBADM authority can remove any task or status record. If an unauthorized user attempts to remove a task or status record, an SQL0551N is returned.

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

## Usage notes

The task is not removed until the unit of work is committed.

The behavior of the task removal depends on how the *name* and *end\_timestamp* arguments are defined:

- If the *end\_timestamp* argument is NULL:
  - If the *name* argument is NULL, all tasks and status records are removed. If one or more tasks are currently running, then the task and associated status records are not removed. In this case, SQL1464W is returned.
  - If the *name* argument is not NULL, the task record that matches *name* is removed. If the specified task is currently running, the task is not removed and SQL20453N is returned. If the specified task is removed, all associated status records are removed.
- If the *end\_timestamp* argument is not NULL:
  - If the *name* argument is NULL, all status records with *end\_timestamp* timestamps less than or equal to *end\_timestamp* are removed. No task records are removed. The procedure will not remove any status records that have a status value of RUNNING.
  - If the *name* argument is not NULL, the status records for the task that matches *name* are removed if their *end\_timestamp* timestamp is less than or equal to *end\_timestamp*. No task records are removed. The procedure will not remove any status records that have a status value of RUNNING.

If a user attempts to remove a task that does not exist, an SQL0204N is returned.

### Example

Remove a backup task called 'DAILY TSM BACKUP':

```
CALL SYSPROC.ADMIN_TASK_REMOVE('DAILY TSM BACKUP', NULL)
```

### ADMIN\_TASK\_STATUS administrative view - Retrieve task status information

The ADMIN\_TASK\_STATUS administrative view retrieves information about the status of task execution in the administrative task scheduler.

The schema is SYSTOOLS.

This view is created the first time the ADMIN\_TASK\_ADD procedure is called.

### Authorization

SELECT or CONTROL privilege on the ADMIN\_TASK\_STATUS administrative view. Unless the database was created with the **RESTRICTIVE** option, SELECT privilege is granted to PUBLIC by default.

When you query the ADMIN\_TASK\_STATUS view, it will only return the task status records that were created by your session authorization ID.

### Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

### Examples

*Example 1:* Request the status of tasks in the scheduler:

```
SELECT * from SYSTOOLS.ADMIN_TASK_STATUS
```

*Example 2:* Format the data in the SQLERRMC column using the SQLERRM function:

```
SELECT TASKID, STATUS, SQLCODE, SQLSTATE, RC,
 VARCHAR(SQLERRM('SQL' || CHAR(ABS(SQLCODE)),
 SQLERRMC, x'FF', 'en_US', 1), 256) AS MSG_TXT
FROM SYSTOOLS.ADMIN_TASK_STATUS
```

### Information returned

Table 52. Information returned by the ADMIN\_TASK\_STATUS administrative view

Column name	Data type	Description
NAME	VARCHAR(128)	The name of the task.
TASKID	INTEGER	The task identifier.

Table 52. Information returned by the ADMIN\_TASK\_STATUS administrative view (continued)

Column name	Data type	Description
STATUS	VARCHAR(10)	The status of the task. Valid values are: <ul style="list-style-type: none"> <li>RUNNING - The task is currently running.</li> <li>COMPLETED - The task has finished running.</li> <li>NOTRUN - An error prevented the scheduler from calling the task's procedure.</li> <li>UNKNOWN - The task started running but an unexpected condition prevented the scheduler from recording the task outcome. This can occur if the system ends abnormally or a power failure happens while the task is running.</li> </ul>
INVOCATION	INTEGER	The current invocation count.
BEGIN_TIME	TIMESTAMP	The time that the task began. <sup>1</sup>  If the STATUS is RUNNING, COMPLETED, or UNKNOWN, this value indicates the time that the task started running.  If the STATUS is NOTRUN, it indicates the time that the task should have started.
END_TIME	TIMESTAMP	The time that the task finished running. <sup>1</sup>  This value will be NULL if the STATUS is RUNNING.  If the STATUS is UNKNOWN, this value is the time the task scheduler detected the task was no longer executing and updated the status table.
AGENT_ID	BIGINT	agent_id - Application handle (agent ID) monitor element
SQLCODE	INTEGER	If the STATUS is COMPLETED, this value indicates the SQLCODE returned by the CALL to the procedure.  If the STATUS is NOTRUN, this value indicates the SQLCODE of the error that prevented the task from running.  If the status is RUNNING or UNKNOWN, this value will be NULL.
SQLSTATE	CHAR(5)	If the STATUS is COMPLETED, this value indicates the SQLSTATE returned by the CALL to the procedure.  If the STATUS is NOTRUN, this value indicates the SQLSTATE of the error that prevented the task from running.  If the status is RUNNING or UNKNOWN, this value will be NULL.
SQLERRMC	VARCHAR(70) FOR BIT DATA	Contains one or more tokens, separated by X'FF', as they appear in the SQLERRMC field of the SQLCA. These tokens are substituted for variables in the descriptions of error conditions  If the STATUS is COMPLETED, this value indicates the SQLERRMC returned by the CALL to the procedure.  If the STATUS is NOTRUN, this value indicates the SQLERRMC of the error that prevented the task from running.  If the status is RUNNING or UNKNOWN, this value will be NULL.
RC	INTEGER	If the STATUS is COMPLETED, this contains the return code from the CALL to the procedure if the procedure had a return code. Otherwise, this will be NULL.

- <sup>1</sup> The BEGIN\_TIME and END\_TIME are based on the database server's time zone. The user is responsible for making adjustments for daylight savings time (DST).

## ADMIN\_TASK\_UPDATE procedure - Update an existing task

The ADMIN\_TASK\_UPDATE procedure updates an administrative task, which is any piece of work that can be encapsulated inside a procedure.

### Syntax

```
►► ADMIN_TASK_UPDATE (—name—, —begin_timestamp—, —end_timestamp—, —————►
►max_invocations—, —schedule—, —options—, —remarks—) —————►◄
```

The schema is SYSPROC.

### Procedure parameters

#### *name*

An input argument of type VARCHAR(128) that specifies the name of an existing task. This argument cannot be NULL.

#### *begin\_timestamp*

An input argument of type TIMESTAMP that specifies the earliest time a task can begin execution. The value of this argument cannot be in the past, and it cannot be later than *end\_timestamp*.

When task execution begins depends on how this parameter and the *schedule* parameter are defined:

- If the *begin\_timestamp* argument is not NULL:
  - If the *schedule* argument is NULL, the task execution begins at *begin\_timestamp*.
  - If the *schedule* argument is not NULL, the task execution begins at the next scheduled time at or after *begin\_timestamp*.
- If the *begin\_timestamp* argument is NULL:
  - If the *schedule* argument is NULL, the task execution begins immediately.
  - If the *schedule* argument is not NULL, the task execution begins at the next scheduled time.

#### *end\_timestamp*

An input argument of type TIMESTAMP that specifies the last time that a task can begin execution. The value of this argument cannot be in the past, and it cannot be earlier than *begin\_timestamp*. If the argument is NULL, the task can continue to execute as scheduled indefinitely.

An executing task will not be interrupted at its *end\_timestamp*.

#### *max\_invocations*

An input argument of type INTEGER that specifies the maximum number of executions allowed for the task. If the argument is NULL, there is no limit to the number of times the task can execute. If the argument is 0, the task will not execute.

This value applies to the schedule if *schedule* is not NULL.

If both *end\_timestamp* and *max\_invocations* are specified, *end\_timestamp* takes precedence. That is, if the *end\_timestamp* timestamp is reached, even though the number of task executions so far has not reached the value of *max\_invocations*, the task will not be executed again.

#### ***schedule***

An input argument of type VARCHAR(1024) that specifies a task execution schedule at fixed points in time. If the argument is NULL, the task is not scheduled at fixed points in time.

The *schedule* string must be specified using the UNIX cron format.

Multiple schedules are not supported.

#### ***options***

An input argument of type VARCHAR(512). This argument must be NULL.

#### ***remarks***

An input argument of type VARCHAR(254) that specifies a description of the task. This is an optional argument that can be set to NULL.

### **Authorization**

One of the following authorizations is required:

- EXECUTE privilege on the ADMIN\_TASK\_UPDATE procedure
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Although the statement authorization ID might allow the procedure to be executed, a task cannot be updated unless the current session authorization ID matches the session authorization ID that was recorded when the task was created. Users with SYSADM, SYSCTRL, SYSMaint, or DBADM can update any existing task. Attempting to update a task that was added by a different user returns SQL0551N.

### **Default PUBLIC privilege**

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

### **Usage notes**

If invalid arguments are passed into the procedure, SQL0171N will be returned. The tokens of the message will indicate which argument is invalid and the name of the procedure.

Changes to the task do not take effect until the unit of work is committed and the scheduler has fetched the updated task definition. Leaving the unit of work uncommitted may delay or prevent the execution of the existing task.

The scheduler checks for updated tasks every 5 minutes. To ensure the task executes as expected, the earliest begin time, as defined by the *begin\_timestamp*, *end\_timestamp* and *schedule* parameters, should be at least 5 minutes after the unit of work commits.

The database must be active on all database partitions to ensure the task can be executed by the scheduler.

The *begin\_timestamp*, *end\_timestamp*, and *schedule* are based on the database server's time zone. Special attention is required when scheduling a task during the transition period of daylight savings time (DST). If the task is scheduled to run 2:01 AM and it is the time of year when the time springs forward, the task will not run as the clock skips from 2:00 AM to 3:00 AM. If the DST change occurs when the database is activated and there is no task scheduled during the DST change time, the first scheduled task after the DST change will not run. Tasks will run normally from the second task onwards. For example, if no task is scheduled between 2:00 AM to 3:00 AM but there is a task at 3:10 AM which is also the first task after DST change, this task will not run. This is independent of the time after the DST change in which the task is scheduled. For example, whether the first task after DST is scheduled one day or one month after the DST change, the task will not run as long as it is the first task after DST change and the database is kept active the entire time. On the other hand, when the time falls back an hour, tasks that were originally scheduled between 2:00 AM and 3:00 AM will execute twice. The user is responsible for making adjustments for daylight savings time to ensure their required behavior.

When a task is updated, the task's internal invocation counter is reset. To illustrate, consider a recurring task with a *max\_invocations* value of 10. If the task executes 3 times, there are 3 corresponding status records in the ADMIN\_TASK\_STATUS output. The entries have INVOCATION values of 1, 2, and 3. Now assume the task creator updates the task. This update will reset the internal invocation counter. The original status records remain in intact. Over time, new status records will be created with INVOCATION values of 1, 2, 3, and so on. The BEGIN\_TIME can be used to distinguish between the original and updated task execution.

## Audit routines and procedures

This grouping of routines and views can be used to work with archive audit logs.

### AUDIT\_ARCHIVE procedure and table function - Archive audit log file

The AUDIT\_ARCHIVE procedure and table function both archive the audit log file for the connected database.

#### Syntax

►►—AUDIT\_ARCHIVE—(—*directory*—,—*member*—)—►►

The schema is SYSPROC.

The syntax is the same for both the procedure and table function.

#### Procedure and table function parameters

##### *directory*

An input argument of type VARCHAR(1024) that specifies the directory where the archived audit file(s) will be written. The directory must exist on the server and the instance owner must be able to create files in that directory. If the argument is null or an empty string, the default directory is used.

*member*

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, NULL or -2 for an aggregate of all members.

## Authorization

Execute privilege on the AUDIT\_ARCHIVE procedure or table function.

## Default PUBLIC privilege

None

## Examples

*Example 1:* Archive the audit log(s) for all members to the default directory using the procedure.

```
CALL SYSPROC.AUDIT_ARCHIVE(NULL, NULL)
```

*Example 2:* Archive the audit log(s) for all members to the default directory using the table function.

```
SELECT * FROM TABLE(SYSPROC.AUDIT_ARCHIVE(' ', -2)) AS T1
```

## Information returned

*Table 53. Information returned by the AUDIT\_ARCHIVE procedure and table function*

Column name	Data type	Description
DBPARTITIONNUM	SMALLINT	dbpartitionnum - Database partition number monitor element
PATH	VARCHAR(1024)	Directory location of archived file.
FILE	VARCHAR(1024)	Name of the archived file.
SQLCODE	INTEGER	The SQLCODE received while attempting to archive file.
SQLSTATE	VARCHAR(5)	The SQLSTATE received while attempting archive file. If SQLSTATE is NULL, the value is zero.
SQLERRMC	VARCHAR(70) FOR BIT DATA	The sqlerrmc received while attempting archive file. If SQLSTATE is NULL, the value is zero.
MEMBER	SMALLINT	member - Database member monitor element

## AUDIT\_DELIM\_EXTRACT - performs extract to delimited file

The AUDIT\_DELIM\_EXTRACT stored procedure performs an extract to a delimited file on archived audit files of the connected database. Specifically, to those archived audit files that have filenames that match the specified mask pattern.

## Syntax

```
►—AUDIT_DELIM_EXTRACT—(—delimiter—,—target_directory—,—source_directory—,—
►—file_mask—,—event_options—)—————►
```

The schema is SYSPROC.

## Procedure parameters

### *delimiter*

An optional input argument of type VARCHAR(1) that specifies the character delimiter to be used in the delimited files. If the argument is null or an empty string, a double quote will be used as the delimiter.

### *target\_directory*

An optional input argument of type VARCHAR(1024) that specifies the directory where the delimited files will be stored. If the argument is null or an empty string, same directory as the *source\_directory* will be used

### *source\_directory*

An optional input argument of type VARCHAR(1024) that specifies the directory where the archived audit log files are stored. If the argument is null or an empty string, the audit default will be used.

### *file\_mask*

An optional input argument of type VARCHAR(1024) is a mask for which files to extract. If the argument is null or an empty string, it will extract from all audit log files in the source directory.

### *event\_options*

An optional input argument of type VARCHAR(1024) that specifies the string defines which events to extract. This matches the same string in the db2audit utility. If the argument is null or an empty string, it will extract all events.

## Authorization

Execute privilege on the AUDIT\_DELIM\_EXTRACT function.

## Default PUBLIC privilege

None

## Examples

**Note:** Audit log files contain a timestamp as part of their naming convention.

*Example 1:* Performs a delimited extract on all audit log files archived on June 18th, 2007 in the default archive directory. This example is extracting just execute events, using a double quote (") character delimiter, and creating or appending the resulting extract files (<category>.del) in the \$HOME/audit\_delim\_extract directory.

```
CALL SYSPROC.AUDIT_DELIM_EXTRACT(NULL, '$HOME/AUDIT_DELIM_EXTRACT', NULL, '%20070618%', 'CATEGORY EXECUTE')
```

## AUDIT\_LIST\_LOGS table function - Lists archived audit log files

The AUDIT\_LIST\_LOGS table function lists the archived audit log files for a database which are present in the specified directory.

## Syntax

►►—AUDIT\_LIST\_LOGS—(—*directory*—)—————►◄

The schema is SYSPROC.

## Procedure parameters

### *directory*

An optional input argument of type VARCHAR(1024) that specifies the directory where the archived audit file(s) will be written. The directory must exist on the server and the instance owner must be able to create files in that directory. If the argument is null or an empty string, then the search default directory is used.

## Authorization

EXECUTE privilege on AUDIT\_LIST\_LOGS table function.

## Default PUBLIC privilege

None

## Examples

*Example 1:* Lists all archived audit logs in the default audit archive directory:

```
SELECT * FROM TABLE(SYSPROC.AUDIT_LIST_LOGS('')) AS T1
```

**Note:** This only lists the logs in the directory for database on which the query is run. Archived files have the format db2audit.db.<dbname>.log.<timestamp>

## Information Returned

Table 54. The information returned for AUDIT\_LIST\_LOGS

Column Name	Data Type	Description
PATH	VARCHAR(1024)	Path location of the archived file.
FILE	VARCHAR(1024)	Filename of the archived file.
SIZE	BIGINT	File size of the archived file.

## Automatic maintenance routines

This grouping of routines and views can be used to work with automatic maintenance policies for a database.

### **AUTOMAINT\_GET\_POLICY procedure - retrieve automatic maintenance policy**

The AUTOMAINT\_GET\_POLICY system stored procedure retrieves the automatic maintenance configuration for the database.

This procedure takes two parameters: the type of automatic maintenance about which to collect information; and a pointer to a BLOB in which to return the configuration information. The configuration information is returned in XML format.

## Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority

- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Syntax

►►AUTOMAINT\_GET\_POLICY(—*policy\_type*—,—*policy*—)—————►►

The schema is SYSPROC.

### Procedure parameters

#### *policy\_type*

An input argument of type VARCHAR(128) that specifies the type of automatic maintenance policy to retrieve. The argument can be one of the following values:

#### **AUTO\_BACKUP**

automatic backup

#### **AUTO\_REORG**

automatic table and index reorganization

#### **AUTO\_RUNSTATS**

automatic table runstats operations

#### **MAINTENANCE\_WINDOW**

maintenance window

#### *policy*

An output argument of type BLOB(2M) that specifies the automatic maintenance settings for the given policy type, in XML format.

### Example

Here is an example of a call to the AUTOMAINT\_GET\_POLICY procedure from within embedded SQL C source code.

- A BLOB variable is declared for the procedure output parameter.
- The procedure is called, specifying automated backup as the type of automatic maintenance policy, and specifying the BLOB variable as the output parameter in which the procedure will return the backup policy for the currently connected database.

```
EXEC SQL BEGIN DECLARE SECTION;
SQL TYPE IS BLOB(2M) backupPolicy;
EXEC SQL END DECLARE SECTION;
```

```
EXEC SQL CALL AUTOMAINT_GET_POLICY('AUTO_BACKUP', :backupPolicy);
```

### **AUTOMAINT\_GET\_POLICYFILE procedure - retrieve automatic maintenance policy**

The AUTOMAINT\_GET\_POLICYFILE system stored procedure retrieves the automatic maintenance configuration for the database.

This procedure takes two parameters: the type of automatic maintenance about which to collect information; and the name of a file in which to return the configuration information. The configuration information is returned in XML format.

## Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

None

## Syntax

►►AUTOMAINT\_GET\_POLICYFILE—(*—policy\_type—*,*—policy\_file\_name—*)————►►

The schema is SYSPROC.

## Procedure parameters

### *policy\_type*

An input argument of type VARCHAR(128) that specifies the type of automatic maintenance policy to retrieve. The argument can be one of the following values:

#### **AUTO\_BACKUP**

automatic backup

#### **AUTO\_REORG**

automatic table and index reorganization

#### **AUTO\_RUNSTATS**

automatic table runstats operations

#### **MAINTENANCE\_WINDOW**

maintenance window

### *policy\_file\_name*

An input argument of type VARCHAR(2048) that specifies the name of the file that is created in the tmp subdirectory of the database instance directory.

**Note:** The file name might be prefixed with a path relative to tmp. If so, the directory should exist, should have permission to create or overwrite the file, and the correct path separator for the database server must be used.

For example:

On UNIX if the instance directory is defined as \$HOME/sql11b. For a policy file named 'policy.xml', the file name will be '\$HOME/sql11b/tmp/policy.xml'

On Windows, the instance directory name can be determined from the values of the **DB2INSTPROF** registry variable and the **DB2INSTANCE** environment variable.

For a policy file named 'policy.xml', if **db2set** gives DB2INSTPROF=C:\DB2PROF and %DB2INSTANCE%=db2, then the file name will be C:\DB2PROF\db2\tmp\policy.xml

### Example

To get the current automatic maintenance settings for backup operations:

```
call sysproc.automaint_get_policyfile('AUTO_BACKUP', 'AutoBackup.xml')
```

This will create an XML file named AutoBackup.xml in the tmp subdirectory under the database instance directory.

### AUTOMAINT\_SET\_POLICY procedure - configure automatic maintenance policy

You can use the AUTOMAINT\_SET\_POLICY system stored procedure to configure automatic maintenance for the database.

This procedure takes two parameters: the type of automatic maintenance to configure; and a BLOB containing XML that specifies the configuration.

To enable the RECLAIM EXTENTS option during the automatic reorganization operations on multidimensional clustering (MDC) or insert time clustering (ITC) tables, you need to specify the “reclaimExtentSizeForTables” attribute to the ReorgOptions element, along with a threshold value in the XML input files.

To enable the RECLAIM EXTENTS option during the automatic reorganization operations on indexes, you need to specify the “reclaimExtentsSizeForIndexObjects” attribute to the ReorgOptions element, along with a threshold value in the XML input files.

**Note:** The threshold specified for “reclaimExtentsSizeForIndexObjects” applies on an index object level. For a nonpartitioned table the value applies to all indexes on the table combined. For a partitioned table the value applies to each nonpartitioned index separately, and to the indexes of each data partition separately.

The threshold values specified for table or index space reclaim is the minimum size, in kilobytes, of reclaimable space in the table or index before an online reorganization to reclaim space is triggered. This threshold value must be 0 or larger. For example, if you specify a value of 1024 KB for the threshold, only objects with 1 MB of reclaimable space or more are considered for automatic reorganization to reclaim space.

### Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

## Syntax

►—AUTOMAINT\_SET\_POLICY—(*—policy\_type—*,*—policy—*)—►

The schema is SYSPROC.

## Procedure parameters

### *policy\_type*

An input argument of type VARCHAR(128) that specifies the type of automatic maintenance policy to configure. The value can be one of:

#### **AUTO\_BACKUP**

automatic backup

#### **AUTO\_REORG**

automatic table and index reorganization

#### **AUTO\_RUNSTATS**

automatic table runstats operations

#### **MAINTENANCE\_WINDOW**

maintenance window

### *policy*

An input argument of type BLOB(2M) that specifies the automatic maintenance policy in XML format. See the sample XML files in the SQLLIB/samples/automaintcfg directory for full details about each element and attribute that can be specified.

## Examples

**Example 1:** To set the current automatic maintenance settings for runstats operations:

```
CALL SYSPROC.AUTOMAINT_SET_POLICY
('AUTO_RUNSTATS',
 BLOB(' <?xml version="1.0" encoding="UTF-8"?>
 <DB2AutoRunstatsPolicy
 xmlns="http://www.ibm.com/xmlns/prod/db2/autonomic/config">
 <RunstatsTableScope><FilterCondition/></RunstatsTableScope>
 </DB2AutoRunstatsPolicy>')
)
```

This replaces the current automatic statistics collection configuration with the new configuration contained in the XML document that is passed as the second parameter to the procedure."

**Example 2:** The automatic reorganization feature can use the new "RECLAIM EXTENTS" option to reorganize multidimensional clustering (MDC) or insert time clustering (ITC) tables. To enable this feature, set the "reclaimExtentSizeForTables" value in the AUTO\_REORG policy:

```
CALL SYSPROC.AUTOMAINT_SET_POLICY
('AUTO_REORG',
 BLOB(' <?xml version="1.0" encoding="UTF-8"?>
 <DB2AutoReorgPolicy
 xmlns="http://www.ibm.com/xmlns/prod/db2/autonomic/config">
 <ReorgOptions dictionaryOption="Keep" indexReorgMode="Online"
 useSystemTempTableSpace="false" reclaimExtentSizeForTables ="1024" >
 </ReorgTableScope>
 </DB2AutoReorgPolicy>
')
```

```

 <FilterClause>TABSCHEMA NOT LIKE 'EMP%'</FilterClause>
 </ReorgTableScope>
</DB2AutoReorgPolicy>')
)

```

There are sample XML input files located in the SQLLIB/samples/automaintcfg directory that you can modify to suit your requirements and then pass the XML content in through the BLOB() scalar function as in the example.

## **AUTOMAINT\_SET\_POLICYFILE procedure - configure automatic maintenance policy**

You can use the AUTOMAINT\_SET\_POLICYFILE system stored procedure to configure automatic maintenance for the database.

This procedure takes two parameters: the type of automatic maintenance to configure; and the name of an XML document that specifies the configuration.

This procedure return the SQL success or SQL error code.

### **Authorization**

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### **Default PUBLIC privilege**

None

### **Syntax**

```

▶▶AUTOMAINT_SET_POLICYFILE(—policy_type—,—policy_file_name—)————▶▶

```

The schema is SYSPROC.

### **Procedure Parameters**

#### *policy\_type*

An input argument of type VARCHAR(128) that specifies the type of automatic maintenance policy to configure. The argument can be one of the following values:

#### ***AUTO\_BACKUP***

automatic backup

#### ***AUTO\_REORG***

automatic table and index reorganization

#### ***AUTO\_RUNSTATS***

automatic table runstats operations

#### ***MAINTENANCE\_WINDOW***

maintenance window

#### *policy\_file\_name*

An input argument of type VARCHAR(2048) that specifies the name of the file that is available in the tmp subdirectory of the database instance directory.

**Note:** When the file name is specified with a relative path, the correct path separator for the database server must be used and the directory and file should exist with read permission.

For example:

On UNIX if the instance directory is defined as \$HOME/sqllib. For a policy file named 'automaint/policy.xml', the file name will be '\$HOME/sqllib/tmp/automaint/policy.xml'

On Windows, the instance directory name can be determined from the values of the **DB2INSTPROF** registry variable and the **DB2INSTANCE** environment variable. For a policy file named 'automaint\policy.xml', if **db2set** gives DB2INSTPROF=C:\DB2PROF and %DB2INSTANCE%=db2, then the file name will be C:\DB2PROF\db2\tmp\automaint\policy.xml

### **Example**

To modify the current automatic maintenance settings for automatic backup:

```
call sysproc.automaint_set_policyfile('AUTO_BACKUP', 'AutoBackup.xml')
```

This will replace the current automatic backup configuration settings with the new configuration contained in the AutoBackup.xml file located in the tmp directory under the database instance directory.

There are sample XML input files located in the SQLLIB/samples/automaintcfg directory which can be used as reference to create policy xml files.

## **Common SQL API procedures**

The common SQL API provides a collection of common-signature and signature-stable stored procedures that are portable across IBM data servers. You can use these stored procedures to create applications that perform a variety of common administrative functions, such as getting and setting configuration parameters, and getting system information.

The stored procedures provide syntactically identical XML parameters and error handling across all data servers to ensure data server version independence. Signature-stability and commonality are achieved by using simple XML documents (with a common DTD) as parameters. Version, platform, and technology differences are expressed through different key value pairs in hierarchical property lists.

### **Common input and output parameters**

The common SQL API stored procedures share a set of input and output parameters.

The following table provides a brief description of these parameters. For more detailed information, see the reference topics about the common SQL API stored procedures.

Table 55. Common SQL API shared input and output parameters

Parameter	Description
<i>major_version</i>	Indicates the document type major version that the caller supports for the XML documents passed as parameters in the procedure.
<i>minor_version</i>	Indicates the document type minor version that the caller supports for the XML documents passed as parameters in the procedure.  The parameters <i>major_version</i> and <i>minor_version</i> are used together to ensure that the caller does not use an XML input document of an incorrect version. The procedure processes all XML documents in the specified <i>major_version</i> and <i>minor_version</i> , or returns an error if a version is not valid. This design supports extensibility in future releases because newer document type versions can be added without affecting existing applications.
<i>requested_locale</i>	Specifies the locale to use to return translated content in the XML documents returned in the <i>xml_output</i> and <i>xml_message</i> parameters. Only values are translated, not key names.
<i>xml_input</i>	Specifies an XML input document that contains input values for the procedure.
<i>xml_filter</i>	Specifies a valid XPath query string that is used to retrieve a single value from an output parameter document.
<i>xml_output</i>	Returns a complete XML output document encoded in UTF-8. Depending on the procedure that is being called, this document might contain configuration parameters and their values, system information, or message text. When the procedure operates in <i>complete mode</i> , this parameter returns an XML document that you can modify and pass back to the procedure as the <i>xml_input</i> parameter. This approach provides a programmatic way to create valid XML input documents.
<i>xml_message</i>	Returns a complete XML output document of type Data Server Message in UTF-8 that provides detailed information about a SQL warning condition.

## Versioning of XML documents

To support extensibility in future releases, the common SQL API stored procedures return XML output documents that include version information.

Whenever the structure of an XML output document changes (for example, when an element is added or removed), the version levels are incremented. Therefore, a procedure might support several versions of the XML output document.

Version information in the XML document is expressed as key-value pairs for document type major version and document type minor version. For example, an XML output document might define the following keys and values in a dictionary element:

```
<key>Document Type Name</key><string>Data Server Configuration Output</string>
<key>Document Type Major Version</key><integer>2</integer>
<key>Document Type Minor Version</key><integer>0</integer>
```

When you call the procedure, you specify the major version and minor version of the XML document that you want to return. The contents of the XML output document will vary depending on the values that you specify.

For example, the GET\_CONFIG procedure retrieves the database and database manager configuration parameters that are set for a particular instance. When this procedure is called with *major\_version* 2 and *minor\_version* 0, it returns an XML document that contains configuration parameters grouped into categories. However, when the same procedure is called with *major\_version* 1 and *minor\_version* 0, it returns an XML document that contains configuration parameters, but they are not grouped into categories.

Likewise, the GET\_MESSAGE procedure retrieves the message text and SQLSTATE for a specified SQLCODE. When this procedure is called with *major\_version* 2 and *minor\_version* 0, it returns an XML document that contains the short text message, long text message, and SQLSTATE for the corresponding SQLCODE. However, when the same procedure is called with *major\_version* 1 and *minor\_version* 0, it returns an XML document that contains only the short text message and SQLSTATE. The long text message is not available in version 1 of the document.

To determine the highest supported document versions for a procedure, specify NULL for *major\_version*, *minor\_version*, and all other input parameters. The procedure returns the highest supported document versions as values in the *major\_version* and *minor\_version* output parameters, and sets the *xml\_output* and *xml\_message* output parameters to NULL.

If you specify non-null values for *major\_version* and *minor\_version*, you must specify supported document versions, or the procedure raises an error (-20457) to indicate that the procedure encountered an unsupported version.

XML input documents can optionally include values for the document type major version and document type minor version. If these values are specified in the XML input document, then the values passed for *major\_version* and *minor\_version* in the procedure call must exactly match the values that are specified in the XML document, or the procedure raises an error (+20458). This behavior ensures that the caller does not specify an unsupported version of the XML input document.

## XML input documents

The XML documents that are passed as input to common SQL API stored procedures share a simple XML format that is based on a common DTD.

The XML input document consists of a set of entries that are common to all stored procedures, and a set of entries that are specific to each stored procedure. The XML input document has the following general structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>Document Type Name</key><string>Data Server Message Input</string>
 <key>Document Type Major Version</key><integer>1</integer>
```

```

 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Document Locale</key><string>en_US</string>
 <key>Complete</key><false/>
 <dict>
 <!-- Document type specific data appears here. -->
 </dict>
 </dict>
</plist>

```

**Important:** XML input documents must be encoded in UTF-8 and contain English characters only.

### Complete mode for returning valid XML input documents:

You can use *complete mode* to create a valid XML document for any common SQL API stored procedure that accepts input. You can then customize the document and pass it back to the procedure.

To run a procedure in complete mode, specify "true" for the Complete key in the input XML document, and pass the following minimal content:

```

<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
 <dict>
 <key>Complete</key><true/>
 </dict>
</plist>

```

Any XML elements that are not required are ignored and are not returned in the output document.

When you run the procedure, a complete XML input document is returned in the *xml\_output* parameter of the stored procedure. The returned XML document includes a document type and a section for all possible required parameters and optional parameters. The returned XML document also includes other entries (such as display names, hints, and the document locale) that are not required, but are typically needed when rendering the document in a client application.

After rendering the XML document and modifying it in a platform-independent way, you can run the same stored procedure and pass in the modified XML document as input.

### XML output documents

The XML documents that are returned as output from common SQL API stored procedures share a common set of entries.

At a minimum, XML documents returned in the *xml\_output* parameter include the following mandatory, key value pairs:

```

<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
 <dict>
 <key>Document Type Name</key>
 <string>Data Server Configuration Output</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Data Server Product Name</key><string>DSN</string>
 <key>Data Server Product Version</key><string>8.1.0.356</string>
 <key>Data Server Major Version</key><integer>8</integer>
 <key>Data Server Minor Version</key><integer>1</integer>
 <key>Data Server Platform</key><string>z/OS</string>
 <key>Document Locale</key><string>en_US</string>
 </dict>

```

```

 <!-- Document type specific data appears here. -->
</dict>
</plist>

```

Entries in the XML output document might be grouped using nested dictionaries. Each entry in the XML output document describes a single piece of information. The entry consists of the value, the display name, and a hint. Optionally, a display unit might be provided. Display name, hint, and display unit are language-sensitive and will be translated to the language specified in the value for the *requested\_locale* parameter (or the default if the requested locale is not yet supported). In general, an entry has a structure similar to the following example:

```

<key>Real Storage Size</key>
<dict>
 <key>Display Name</key><string>Real Storage Size</string>
 <key>Value</key><integer>2048</integer>
 <key>Display Unit</key><string>MB</string>
 <key>Hint</key><string>Size of actual real storage online</string>
</dict>

```

IBM data servers have a common parameter document that includes some keywords that are applicable to all data servers, and others that are data server specific. Whenever a data server adds or removes a new keyword, the version number (for all data servers) is incremented. Depending on the change, the major version number might be increased and the minor version number set to 0 (zero), or only the minor version number might be incremented.

XML output documents are generated in UTF-8 and contain English characters only.

### **XPath expressions for filtering output:**

You can use an XPath expression to filter the XML output returned by a common SQL API stored procedure.

To filter the output, specify a valid XPath query string in the *xml\_filter* parameter of the procedure. The following restrictions apply to the XPath expression that you specify:

- The XPath expression must reference a single value.
- The XPath expression must always be absolute from the root node. For example, the following path expressions are allowed: */*, *nodename*, *..*, and *...*. The following expressions are not allowed: *//* and *@*.
- The only predicates allowed are *[path='value']* and *[n]*.
- The only axis allowed is *following-sibling*.
- The XPath expression must end with one of the following, and, if necessary, be appended with the predicate [1]: *following-sibling::string*, *following-sibling::data*, *following-sibling::date*, *following-sibling::real*, or *following-sibling::integer*.
- Unless the axis is found at the end of the XPath expression, it must be followed by a *::dict*, *::string*, *::data*, *::date*, *::real*, or *::integer*, and if necessary, be appended with the predicate [1].
- The only supported XPath operator is *=*.
- The XPath expression cannot contain a function, namespace, processing instruction, or comment.

**Tip:** If the stored procedure operates in *complete mode*, do not apply filtering, or a SQLCODE (+20458) is raised.

For better control over processing the XML document returned in the *xml\_output* parameter, you can use the XMLPARSE function available with Db2 pureXML®.

### Example

The following XPath expression selects the value for the Data Server Product Version key from an XML output document:

```
/plist/dict/key[.='Data Server Product Version']/following-sibling::string[1]
```

The procedure returns the string 8.1.0.356 in the *xml\_output* parameter. Therefore, the procedure call returns a single value rather than an XML document.

### XML message documents

When a common SQL API stored procedure encounters an internal processing error or invalid parameter, the data server returns a SQLCODE and the corresponding SQL message to the caller.

When this occurs, the procedure returns an XML message document in the *xml\_message* parameter that contains more detailed information about the warning situation.

The XML message document has the following general structure:

```
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>Document Type Name</key><string>Data Server Message</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Data Server Product Name</key><string>QDB2/AIX64</string>
 <key>Data Server Product Version</key><string>9.5.0.3</string>
 <key>Data Server Major Version</key><integer>9</integer>
 <key>Data Server Minor Version</key><integer>5</integer>
 <key>Data Server Platform</key><string>AIX 64BIT</string>
 <key>Document Locale</key><string>en_US</string>
 <key>Short Message Text</key>
 <dict>
 <key>Value</key><string>
 <!-- Additional description of warning appears here. --></string>
 <key>Hint</key><string></string>
 </dict>
</dict>
</plist>
```

XML message documents are generated in UTF-8 and contain English characters only.

### Example

In the following example, a call to the GET\_MESSAGE procedure results in an SQL warning:

```
db2 "CALL SYSPROC.GET_MESSAGE(NULL,NULL,'en_US',NULL,NULL,?,?)"
SQL20458W The procedure "SYSPROC.GET_MESSAGE" has encountered
an internal parameter processing error in parameter "3".
The value for parameter "7" contains further information about
the error. SQLSTATE=01H54
```

The XML document that is returned in parameter 7 (*xml\_message*) contains the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>Document Type Name</key><string>Data Server Message</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Data Server Product Name</key><string>QDB2/AIX64</string>
 <key>Data Server Product Version</key><string>9.5.0.3</string>
 <key>Data Server Major Version</key><integer>9</integer>
 <key>Data Server Minor Version</key><integer>5</integer>
 <key>Data Server Platform</key><string>AIX 64BIT</string>
 <key>Document Locale</key><string>en_US</string>
 <key>Short Message Text</key>
 <dict>
 <key>Value</key><string>If parameters 1 and 2 are set to NULL, all
 other input parameters must be set to NULL as well, but the value
 of parameter "3" is not NULL. </string>
 <key>Hint</key><string></string>
 </dict>
</dict>
</plist>
```

The value for the Short Message Text key provides additional information about the warning.

## CANCEL\_WORK procedure - Cancel work

The CANCEL\_WORK stored procedure cancels either a specific activity (for example, a SQL statement), or all activity for a connected application.

To cancel a specific activity, you pass in the application handle, unit of work ID, and activity ID for the activity that you want to cancel. To cancel all activity for a connected application, you pass in the application handle. Any changes associated with the cancelled activity are rolled back.

## Syntax

```
►►—CANCEL_WORK—(—major_version—,—minor_version—,—requested_locale—,——————►
►—xml_input—,—xml_filter—,—xml_output—,—xml_message—)——————►►
```

The schema is SYSPROC.

## Procedure parameters

### major\_version

An input and output argument of type INTEGER that indicates the major document version. On input, this argument indicates the major document version that the caller supports for the XML documents passed as parameters in the procedure (see the parameter descriptions for *xml\_input*, *xml\_output*, and *xml\_message*). The procedure processes all XML documents in the specified version, or returns an error (+20458) if the version is not valid. On output, this parameter specifies the highest major document version that is supported by the procedure. To determine the highest supported document version, specify NULL for this input parameter and all other required parameters.

**Supported versions:** 1

### ***minor\_version***

An input and output argument of type INTEGER that indicates the minor document version. On input, this argument specifies the minor document version that the caller supports for the XML documents passed as parameters for this procedure (see the parameter descriptions for *xml\_input*, *xml\_output*, and *xml\_message*). The procedure processes all XML documents in the specified version, or returns an error if the version is not valid. On output, this parameter indicates the highest minor document version that is supported for the highest supported major version. To determine the highest supported document version, specify NULL for this input parameter and all other required parameters.

**Supported versions:** 0

### ***requested\_locale***

An input argument of type VARCHAR(33) that specifies a locale. If the specified language is supported on the server, translated content is returned in the *xml\_output* and *xml\_message* parameters. Otherwise, content is returned in the default language. Only the language and possibly the territory information is used from the locale. The locale is not used to format numbers or influence the document encoding. For example, key names and values are not translated. The only translated portion of the XML output and XML message documents are the text for hint, display name, and display unit of each entry. The caller should always compare the requested language to the language that is used in the XML output document (see the document locale entry in the XML output document).

Currently, the only supported value for *requested\_locale* is en\_US.

### ***xml\_input***

An input argument of type BLOB(32MB) that specifies an XML input document (encoded in UTF-8) that contains input values for the procedure.

For this procedure, the XML input document must specify an application handle. If you want to cancel a specific activity, the XML input document must also specify optional parameters that identify a unit of work ID and an activity ID. A complete XML input document for this stored procedure looks something like the following document:

```
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>Document Type Name</key><string>Data Server Cancel Work Input</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Required Parameters</key>
 <dict>
 <key>Application Handle</key>
 <dict>
 <key>Display name</key><string>Application Handle</string>
 <key>Value</key><integer>10</integer>
 <key>Hint</key>
 <string>
 Numeric value equivalent to the application handle to be cancelled
 </string>
 </dict>
 </dict>
 <key>Optional Parameters</key>
 <dict>
 <key>Unit Of Work Id</key>
 <dict>
 <key>Display Name</key><string>Unit Of Work Id</string>
 <key>Value</key><integer>20</integer>
 </dict>
 </dict>
</dict>
</plist>
```

```

 <key>Hint</key>
 <string>
 Numeric value that specifies the unit of work id of the activity
 that is to be cancelled
 </string>
 </dict>
 <key>Activity Id</key>
 <dict>
 <key>Display Name</key><string>Activity Id</string>
 <key>Value</key><integer>10</integer>
 <key>Hint</key>
 <string>
 Numeric value equivalent to the activity id to be cancelled
 </string>
 </dict>
</dict>
</dict>
</plist>

```

If you specify the application handle of the application where the stored procedure is running, the procedure returns a warning (SQL20458).

#### ***xml\_filter***

An input argument of type BLOB(4K) that specifies a valid XPath query string. Use a filter when you want to retrieve a single value from an XML output document. For more information, see the topic that describes XPath filtering.

The following example selects the value for the Data Server Product Version from the XML output document: /plist/dict/key[.='Data Server Product Version']/following-sibling::string. If the key is not followed by the specified sibling, an error is returned.

#### ***xml\_output***

An output parameter of type BLOB(32MB) that returns a complete XML output document in UTF-8. If a filter is specified, this parameter returns a string value. If the stored procedure is unable to return a complete output document (for example, if a processing error occurs that results in an SQL warning or error), this parameter is set to NULL.

The XML output is determined by the values that you specify for *major\_version* and *minor\_version*:

Major version	Minor version	xml_output value
NULL	NULL	NULL
1	0	The status of the activity that the procedure attempted to cancel.

When the procedure operates in *complete mode*, this parameter returns an XML document that you can modify and pass back to the procedure as the *xml\_input* parameter. This approach provides a programmatic way to create valid XML input documents. For more information, see the topic about complete mode.

#### ***xml\_message***

An output parameter of type BLOB(64K) that returns a complete XML output document of type Data Server Message in UTF-8 that provides detailed information about a SQL warning condition. This document is returned when a call to the procedure results in a SQL warning, and the warning message indicates that additional information is returned in the XML message output

document. If the warning message does not indicate that additional information is returned, then this parameter is set to NULL.

## Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

## Example

*Example 1:* Return the highest supported version of the procedure.

```
db2 "call sysproc.cancel_work(null,null,null,null,null,?,?)"
```

The following is an example of output from this query:

Value of output parameters

-----  
Parameter Name : MAJOR\_VERSION  
Parameter Value : 1

Parameter Name : MINOR\_VERSION  
Parameter Value : 0

Parameter Name : XML\_OUTPUT  
Parameter Value : -

Parameter Name : XML\_MESSAGE  
Parameter Value : -

Return Status = 0

*Example 2:* Cancel a specific activity.

```
db2 "call sysproc.cancel_work(1,0,'en_US',blob(
<?xml version='1.0' encoding='UTF-8'?>
<plist version='1.0'>
<dict>
 <key>Document Type Name</key><string>Data Server Cancel Work Input</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Required Parameters</key>
 <dict>
 <key>Application Handle</key>
 <dict>
 <key>Display name</key><string>Application Handle</string>
 <key>Value</key><integer>1</integer>
 <key>Hint</key>
 <string>
 Numeric value equivalent to the application handle to be cancelled
 </string>
 </dict>
 </dict>
 <key>Optional Parameters</key>
 <dict>
```

```

<key>Unit Of Work Id</key>
<dict>
 <key>Display Name</key><string>Unit Of Work Id</string>
 <key>Value</key><integer>2</integer>
 <key>Hint</key>
 <string>
 Numeric value that specifies the unit of work id of the activity
 that is to be cancelled
 </string>
</dict>
<key>Activity Id</key>
<dict>
 <key>Display Name</key><string>Activity Id</string>
 <key>Value</key><integer>3</integer>
 <key>Hint</key>
 <string>
 Numeric value equivalent to the activity id to be cancelled
 </string>
</dict>
</dict>
</dict>
</plist>) ,null,?,?)"
```

The following is an example of output from this query:

Value of output parameters

```

Parameter Name : MAJOR_VERSION
Parameter Value : 1
```

```

Parameter Name : MINOR_VERSION
Parameter Value : 0
```

```

Parameter Name : XML_OUTPUT
Parameter Value : x'3C3F78...'
```

```

Parameter Name : XML_MESSAGE
Parameter Value : -
```

Return Status = 0

If the CANCEL\_WORK procedure is able to cancel the activity, the XML output document contains the following content:

```

<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict><key>Document Type Name</key><string>Data Server Cancel Work Output</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Data Server Product Name</key><string>QDB2/AIX64</string>
 <key>Data Server Product Version</key><string>9.7.0.0</string>
 <key>Data Server Major Version</key><integer>9</integer>
 <key>Data Server Minor Version</key><integer>7</integer>
 <key>Data Server Platform</key><string>AIX 64BIT</string>
 <key>Document Locale</key><string>en_US</string>
 <key>Successful Cancel Work Message</key>
 <dict>
 <key>Display Name</key><string>Successful Cancel Work Message</string>
 <key>Value</key><string>The activity has been cancelled successfully</string>
 <key>Hint</key><string></string>
 </dict>
</dict>
</plist>
```

*Example 2: Cancel the application.*

```

db2 "call sysproc.cancel_work(1,0,'en_US,blob(
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>Document Type Name</key><string>Data Server Cancel Work Input</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Required Parameters</key>
 <dict>
 <key>Application Handle</key>
 <dict>
 <key>Display name</key><string>Application Handle</string>
 <key>Value</key><integer>101</integer>
 <key>Hint</key>
 <string>
 Numeric value equivalent to the application handle to be cancelled
 </string>
 </dict>
 </dict>
</dict>
</plist>),null,?,?)"
```

The following is an example of output from this query:

Value of output parameters

-----  
Parameter Name : MAJOR\_VERSION  
Parameter Value : 1

Parameter Name : MINOR\_VERSION  
Parameter Value : 0

Parameter Name : XML\_OUTPUT  
Parameter Value : x'3C3F78...'

Parameter Name : XML\_MESSAGE  
Parameter Value : -

Return Status = 0

If the CANCEL\_WORK procedure is able to cancel the application, the XML output document contains the following content:

```

<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>Document Type Name</key><string>Data Server Cancel Work Output</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Data Server Product Name</key><string>QDB2/AIX64</string>
 <key>Data Server Product Version</key><string>9.7.0.0</string>
 <key>Data Server Major Version</key><integer>9</integer>
 <key>Data Server Minor Version</key><integer>7</integer>
 <key>Data Server Platform</key><string>AIX 64BIT</string>
 <key>Document Locale</key><string>en_US</string>
 <key>Successful Cancel Work Message</key>
 <dict>
 <key>Display Name</key><string>Successful Cancel Work Message</string>
 <key>Value</key>
 <string>The application has been cancelled successfully</string>
 <key>Hint</key><string></string>
 </dict>
</dict>
</plist>
```

*Example 3:* Specify a filter to return the value of a successful cancel work message.

```

db2 "call sysproc.cancel_work(1,0,'en_US,blob(
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>Document Type Name</key><string>Data Server Cancel Work Input</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Required Parameters</key>
 <dict>
 <key>Application Handle</key>
 <dict>
 <key>Display name</key><string>Application Handle</string>
 <key>Value</key><integer>101</integer>
 <key>Hint</key>
 <string>
 Numeric value equivalent to the application handle to be cancelled
 </string>
 </dict>
 </dict>
</dict>
</plist>),blob('/plist/dict/key[.="Successful Cancel Work Message"]
/following-sibling::dict[1]/key[.="Value"]
/following-sibling::string[1]'),'?,?)'"

```

The following is an example of output from this query:

```

Value of output parameters

Parameter Name : MAJOR_VERSION
Parameter Value : 1

Parameter Name : MINOR_VERSION
Parameter Value : 0

Parameter Name : XML_OUTPUT
Parameter Value : x'3C3F78...'

Parameter Name : XML_MESSAGE
Parameter Value : -

Return Status = 0

```

The following value is returned for *xml\_output*:

```
"The application has been cancelled successfully"
```

## DESIGN\_ADVISOR - retrieve design advisor recommendations

The DESIGN\_ADVISOR procedure retrieves design advisor recommendations from a database server.

### Syntax

```

►►—DESIGN_ADVISOR—(—major_version—,—minor_version—,—requested_locale—,——————►
►—xml_input—,—xml_filter—,—xml_output—,—xml_message—)——————►◄

```

The schema is SYSPROC.

### Procedure parameters

#### *major\_version*

An input and output argument of type INTEGER that indicates the major document version. On input, this argument indicates the major document version that the caller supports for the XML documents passed as parameters

in the procedure (see the parameter descriptions for *xml\_input*, *xml\_output*, and *xml\_message*). The procedure processes all XML documents in the specified version, or returns an error (+20458) if the version is not valid. On output, this parameter specifies the highest major document version that is supported by the procedure. To determine the highest supported document version, specify NULL for this input parameter and all other required parameters.

If the *xml\_input* parameter specifies a Document Type Major Version key and the *major\_version* parameter is not NULL, then the *major\_version* parameter takes precedence.

**Supported versions:** 1

***minor\_version***

An input and output argument of type INTEGER that indicates the minor document version. On input, this argument specifies the minor document version that the caller supports for the XML documents passed as parameters for this procedure (see the parameter descriptions for *xml\_input*, *xml\_output*, and *xml\_message*). The procedure processes all XML documents in the specified version, or returns an error if the version is not valid. On output, this parameter indicates the highest minor document version that is supported for the highest supported major version. To determine the highest supported document version, specify NULL for this input parameter and all other required parameters.

If the *xml\_input* parameter specifies a Document Type Minor Version key and the *minor\_version* parameter is not NULL, then the *minor\_version* parameter takes precedence.

**Supported versions:** 0

***requested\_locale***

An input argument of type VARCHAR(33) that specifies a locale. If the specified language is supported on the server, translated content is returned in the *xml\_output* and *xml\_message* parameters. Otherwise, content is returned in the default language. Only the language, and possibly the territory information, is used from the locale. The locale is not used to format numbers or influence the document encoding. For example, key names and values are not translated. The only translated portion of the XML output and XML message documents is the error message text. The caller should always compare the requested language to the language that is used in the XML output document (see the document locale entry in the XML output document).

Currently, the only supported value for *requested\_locale* is en\_US.

***xml\_input***

An input argument of type BLOB(32M) that specifies a PLIST XML input string.

***xml\_filter***

An input argument of type BLOB(4K). This parameter is reserved for future use.

***xml\_output***

An output parameter of type BLOB(12K) that returns a PLIST XML output string.

***xml\_message***

An output parameter of type BLOB(64K) that returns a complete XML output

document of type Data Server Message, in UTF-8 encoding. This document provides detailed information about an SQL warning condition.

### Authorization

- Read access to the database.
- Read and write access to the explain tables of the currently connected schema or the SYSTOOLS schema.
- If materialized query tables (MQTs) are used, you must have CREATE TABLE authorization, and read and write access to the MQTs
- EXECUTE privilege on the DESIGN\_ADVISOR function.

### Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

### Information returned

This information is always returned unless an error is generated.

*Table 56. Information returned by the DESIGN\_ADVISOR - retrieve design advisor recommendations table function*

Column name	Data type	Description
SCHEMA	VARCHAR(128)	Schema name of the object or the proposed object to which this recommendation applies.
NAME	VARCHAR(128)	Name of the object or the proposed object to which this recommendation applies.
EXISTS	CHAR(1)	Indicates that the object exists.
RECOMMENDATION	VARCHAR(8)	Indicates the recommendation type. Valid values are: <ul style="list-style-type: none"> <li>• M for MQT</li> <li>• P for database partitioning</li> <li>• C for MDC</li> <li>• I for index</li> <li>• D if the object is not used by the given workload and can be considered for deletion.</li> </ul> The result can be any combination of these values. For example "MC" indicates that the table is recommended as an MQT and an MDC table.
BENEFIT	DOUBLE	Estimated benefit, in timerons, of the proposed object or modification to the proposed object of the query. For base tables or MQTs that have MDC or partitioning recommendations, this value is NULL.

Table 56. Information returned by the *DESIGN\_ADVISOR* - retrieve design advisor recommendations table function (continued)

Column name	Data type	Description
OVERHEAD	DOUBLE	Estimated cost, in timerons, to maintain either the proposed object or the modification to the proposed object. Indexes are ranked with the formula $BENEFIT - OVERHEAD$ . MQTs are ranked with $BENEFIT - (0.5 * OVERHEAD)$ . For base tables or MQTs that have MDC or partitioning recommendations, this value is NULL.
STATEMENT_NO	INTEGER	Statement number referred to by this recommendation. Reflects the statement number in the ADVISE_WORKLOAD table. When a recommendation applies to multiple statements, only one row is returned for each statement.
DISKUSE	DOUBLE	Estimated size, in MB, of either the recommended object or the result of modifications to the current object on disk.

## Usage notes

### XML\_INPUT options

Table 57. XML\_INPUT options

Key name	Optional	Data type	Description
MAJOR_VERSION	Y	INTEGER	XML_OUTPUT schema major version supported by the client as input. If the procedure parameter of the same name is provided, it is used. Otherwise, this value is retrieved and required in XML_INPUT.
MINOR_VERSION	Y	INTEGER	XML_OUTPUT schema minor version supported by the client as input. If the procedure parameter of the same name is provided, it is used. Otherwise, this value is retrieved and required in XML_INPUT.
REQUESTED_LOCALE	Y	STRING	If the locale is supported at the server, the error messages are returned in the requested locale. If the locale is unsupported or invalid, the data is returned in the default locale of the server. If the procedure parameter of the same name is provided, it is used. Otherwise, this value is retrieved and required in XML_INPUT.
CMD_OPTIONS	N	STRING	List of arguments as accepted by the <b>db2adv</b> command. See the “Usage notes” section for a list of differences between the command-line parameters accepted by the <b>db2adv</b> command and this procedure.

Table 57. XML\_INPUT options (continued)

Key name	Optional	Data type	Description
USER_TEMP_TSPACE	Y	STRING	The name of a USER TEMPORARY TABLESPACE where a declared global temporary table (DGTG) can be created to hold the result set. If no name is provided, fenced process memory is used instead. The supplied table space must exist, be writeable by the caller, and have enough space for the entire result set. The DGTG exists and uses system resources until the application disconnects. Contents are deleted each time to save space and because the output is non-deterministic.

**Note:** Special characters like "&", "<", ">", "'" (single quotation mark), and '"' (double quotation mark) can be specified by their corresponding HTML entities of "&";", "&lt;", "&gt;", "&apos;", and "&quot;".

#### XML\_OUTPUT description

The XML\_OUTPUT document is always returned in a UTF-8 code page. Code page conversion is done for database identifiers, objects, and other possible non-UTF-8 characters. Special characters (see XML\_INPUT options for a list) are translated as well.

Table 58. XML\_OUTPUT description

Key name	Data type	Description
Document Type Name	STRING	Always returns the string "Data Server Message Output"
MAJOR_VERSION	INTEGER	Document version. Currently, the only return value is 1.
MINOR_VERSION	INTEGER	Document sub-version. Currently, the only return value is 0.
NUM_OUTPUT_ROWS	INTEGER	Number of rows returned in the result set.
ADVISE_START_TIME	STRING	Timestamp when the advisor began working. Equivalent to the ADVISE_INSTANCE.START_TIME column.
WORKLOAD_NAME	STRING	Name of the workload used by the advisor. Equivalent to the ADVISE_WORKLOAD.WORKLOAD_NAME column.
ADVISE_SCHEMA	STRING	Name of the explain/advisor table schema. This schema is used to read from and write to the ADVISE_WORKLOAD.ADVISE_INSTANCE and other explain/advisor tables.
TOTAL_DISK	STRING	Total initial disk space, in MB, needed if all recommended objects are to be created.
TOTAL_DISK_UPPER_BOUND	INTEGER	Upper bound limit for total disk space used when evaluating solution options, in MB.
ORIG_TOTAL_COST	STRING	Total cost, in timerons, without recommendations.

Table 58. XML\_OUTPUT description (continued)

Key name	Data type	Description
NEW_TOTAL_COST	STRING	Total cost, in timerons, with recommendations.
NUM_SOLUTIONS_EVAL	INTEGER	Number of solutions considered and evaluated by the advisor.

### Difference between db2adviz command-line parameters and DESIGN\_ADVISOR

These options are not allowed because only the current database connection is being used by the procedure:

#### **-[db | d]**

The database name. The current database connection is used.

#### **-[user | a]**

The username to connect with (and optionally, the user password). In DESIGN\_ADVISOR, this option is replaced by the SESSION\_USER special register.

#### **-[password | x]**

This parameter indicates that the password is read from the standard input. It is not used in DESIGN\_ADVISOR.

For file and directory locations, supply absolute path names whenever possible, to ensure a consistent behavior on different database server installations. Files and directories need to be readable (-file) or writeable (-script) by the fenced user ID on Linux/UNIX, or the DB2USERS group on Windows.

When the command-line options **-file** or **-script** are used, the statements are inserted into the ADVISE\_WORKLOAD table for later reference with a unique workload name.

### Clarification of the different schemas used by db2adviz

#### **Explain/advisor table schema name**

The explain/advisor table schema name used by the DESIGN\_ADVISOR procedure is defined by the CURRENT\_USER special register. This special register defaults to the currently connected user. If the explain/advisor tables are not found through the user ID defined in the previous two options, then the SYSTOOLS schema is used.

#### **Recommended objects schema name**

The schema name for recommended objects is optionally defined using the **-[schema | n]** command-line option. If no name is provided, the value of the SESSION\_USER special register is used by default.

#### **Default workload schema name**

The schema name for the default workload is optionally defined using the **-q** command-line option. If no name is provided, the value of the SESSION\_USER special register is used by default.

### Examples

*Example 1:* An example of an XML\_INPUT:

*Example 2:* An example of an XML\_OUTPUT:

*Example 3:* An example of an XML\_MESSAGE:

*Example 4:* A full initialization and procedure call, with the resulting XML outputs.

```
db2 "INSERT INTO ADVISE_WORKLOAD values('workload 1', 0,
'SELECT COUNT(*) FROM EMPLOYEE', ' ', ' ', 100, 0, 0, 0, 0, ' ')"

db2 "INSERT INTO ADVISE_WORKLOAD values('workload 1', 1,
'SELECT * FROM EMPLOYEE WHERE LASTNAME='HAAS' ', ' ', 100, 0, 0, 0, 0, ' ')"

db2 "INSERT INTO ADVISE_WORKLOAD values('workload 1', 2,
'SELECT AVG(BONUS), AVG(SALARY) FROM EMPLOYEE GROUP BY WORKDEPT
ORDER BY WORKDEPT', ' ', 1, 0, 0, 0, 0, ' ')"
```

```

<key>REQUESTED_LOCALE</key><string>en_US</string>
<key>CMD_OPTIONS</key><string>-i "
/home/dricard/prog/adv spaces! sp/cli/db2advis.in"; -t 5</string>
</dict>
</plist>') , NULL, ?, ?)"

```

The value of the output parameters:

```

Parameter Name : MAJOR_VERSION
Parameter Value : 1

```

```

Parameter Name : MINOR_VERSION
Parameter Value : 0

```

```

Parameter Name : XML_OUTPUT
Parameter Value : x'
<plist version="1.0">
<dict>
<key>Document Type Name</key><string>Data Server Message Output</string>
<key>MAJOR_VERSION</key><integer>1</integer>
<key>MINOR_VERSION</key><integer>0</integer>
<key>NUM_OUTPUT_ROWS</key><integer>3</integer>
<key>NUM_RESULT_SETS</key><integer>1</integer>
<key>ADVISE_START_TIME</key><string>2011-03-10-14.22.51.707742</string>
<key>WORKLOAD_NAME</key><string>MYWORKLOAD</string>
<key>ADVISE_SCHEMA</key><string>MYSCHEMA</string>
<key>TOTAL_DISK</key><string>0.076</string>
<key>TOTAL_DISK_UPPER_BOUND</key><string>33.377</string>
<key>ORIG_TOTAL_COST</key><string>28434.0000</string>
<key>NEW_TOTAL_COST</key><string>11108.0000</string>
<key>NUM_SOLUTIONS_EVAL</key><string>31</string>
</dict>
</plist>'
Parameter Name : XML_MESSAGE
Parameter Value : -

```

The values of the result set:

CREATOR	NAME	EXISTS	RECOMMENDATION	...
DRICARD	XEMP2	Y	I	...
DRICARD	IDX1103211528140	N	I	...
DRICARD	IDX1103211529540	N	I	...

Output from the result set continued:

BENEFIT	OVERHEAD	STMTNO	DISKUSE
+0.000000000000000E+000	+0.000000000000000E+000	0	+0.000000000000000E+000
+1.720000000000000E+004	+0.000000000000000E+000	1	+0.063500000000000E+000
+1.260000000000000E+002	+0.000000000000000E+000	2	+1.271900000000000E-002

3 record(s) selected.

## GET\_CONFIG procedure - Get configuration data

The GET\_CONFIG stored procedure retrieves the database configuration, database manager configuration, and registry variables that are set for a particular instance.

In a partitioned database environment, this procedure retrieves database configuration and registry variable settings from all partitions.

## Syntax

```

➤➤ GET_CONFIG (—major_version—, —minor_version—, —requested_locale—, —————➤

```

►*xml\_input*—,—*xml\_filter*—,—*xml\_output*—,—*xml\_message*—)—————►

The schema is SYSPROC.

## Procedure parameters

### ***major\_version***

An input and output argument of type INTEGER that indicates the major document version. On input, this argument indicates the major document version that the caller supports for the XML documents passed as parameters in the procedure (see the parameter descriptions for *xml\_input*, *xml\_output*, and *xml\_message*). The procedure processes all XML documents in the specified version, or returns an error (+20458) if the version is not valid. On output, this parameter specifies the highest major document version that is supported by the procedure. To determine the highest supported document version, specify NULL for this input parameter and all other required parameters.

**Supported versions:** 1 and 2

### ***minor\_version***

An input and output argument of type INTEGER that indicates the minor document version. On input, this argument specifies the minor document version that the caller supports for the XML documents passed as parameters for this procedure (see the parameter descriptions for *xml\_input*, *xml\_output*, and *xml\_message*). The procedure processes all XML documents in the specified version, or returns an error if the version is not valid. On output, this parameter indicates the highest minor document version that is supported for the highest supported major version. To determine the highest supported document version, specify NULL for this input parameter and all other required parameters.

**Supported versions:** 0

### ***requested\_locale***

An input argument of type VARCHAR(33) that specifies a locale. If the specified language is supported on the server, translated content is returned in the *xml\_output* and *xml\_message* parameters. Otherwise, content is returned in the default language. Only the language and possibly the territory information is used from the locale. The locale is not used to format numbers or influence the document encoding. For example, key names and values are not translated. The only translated portion of the XML output and XML message documents are the text for hint, display name, and display unit of each entry. The caller should always compare the requested language to the language that is used in the XML output document (see the document locale entry in the XML output document).

Currently, the only supported value for *requested\_locale* is en\_US.

### ***xml\_input***

Currently, this procedure accepts no input. You must specify NULL for this parameter, or an error (+20458) is raised to indicate that the input is not valid.

### ***xml\_filter***

An input argument of type BLOB(4K) that specifies a valid XPath query string. Use a filter when you want to retrieve a single value from an XML output document. For more information, see the topic that describes XPath filtering.

The following example selects the value for the data server product version from the XML output document: `/plist/dict/key[.='Data Server Product Version']/following-sibling::string`. If the key is not followed by the specified sibling, an error is returned.

#### ***xml\_output***

An output parameter of type BLOB(32MB) that returns a complete XML output document in UTF-8. If a filter is specified, this parameter returns a string value. If the stored procedure is unable to return a complete output document (for example, if a processing error occurs that results in an SQL warning or error), this parameter is set to NULL.

The XML output is determined by the values that you specify for *major\_version* and *minor\_version*:

Major version	Minor version	<i>xml_output</i> value
NULL	NULL	NULL
1	0	Database manager and database configuration parameters and registry variables, including their values.
2	0	Database manager and database configuration parameters grouped into categories. For each parameter, indicates whether the parameter can be updated. Also returns registry variables and the values set for the instance.

When the procedure operates in *complete mode*, this parameter returns an XML document that you can modify and pass back to the procedure as the *xml\_input* parameter. This approach provides a programmatic way to create valid XML input documents. For more information, see the topic about complete mode.

#### ***xml\_message***

An output parameter of type BLOB(64K) that returns a complete XML output document of type Data Server Message in UTF-8 that provides detailed information about a SQL warning condition. This document is returned when a call to the procedure results in a SQL warning, and the warning message indicates that additional information is returned in the XML message output document. If the warning message does not indicate that additional information is returned, then this parameter is set to NULL.

### **Authorization**

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

### Example

*Example 1:* Return the latest version of the procedure.

```
db2 "call sysproc.get_config(null,null,null,null,null,?,?)"
```

The following is an example of output from this query:

Value of output parameters

-----  
Parameter Name : MAJOR\_VERSION  
Parameter Value : 2

Parameter Name : MINOR\_VERSION  
Parameter Value : 0

Parameter Name : XML\_OUTPUT  
Parameter Value : -

Parameter Name : XML\_MESSAGE  
Parameter Value : -

Return Status = 0

*Example 2:* Return database and database manager configuration parameters, grouped into categories.

```
db2 "call sysproc.get_config(2,0,'en_US',null, null, ?,?)"
```

The following is an example of output from this query:

Value of output parameters

-----  
Parameter Name : MAJOR\_VERSION  
Parameter Value : 2

Parameter Name : MINOR\_VERSION  
Parameter Value : 0

Parameter Name : XML\_OUTPUT  
Parameter Value : x'3C3F78.....'

Parameter Name : XML\_MESSAGE  
Parameter Value : -

Return Status = 0

The XML output document contains the following content:

```
<plist version="1.0">
<dict>
 <key>Document Type Name</key><string>Data Server Configuration Output</string>
 <key>Document Type Major Version</key><integer>2</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Data Server Product Name</key><string>QDB2/AIX64</string>
 <key>Data Server Product Version</key><string>9.7.0.0</string>
 <key>Data Server Major Version</key><integer>9</integer>
 <key>Data Server Minor Version</key><integer>7</integer>
 <key>Data Server Platform</key><string>AIX 64BIT</string>
 <key>Document Locale</key><string>en_US</string>
 <key>Database Manager Configuration Parameter Settings</key>
 <dict>
```

```

<key>Display Name</key>
<string>Database Manager Configuration Parameter Settings</string>
<key>Application</key>
<dict>
 <key>Display Name</key><string>Application</string>
 <key>agentpri</key>
 <dict>
 <key>Display Name</key><string>agentpri</string>
 <key>Parameter Value</key>
 <dict>
 <key>Display Name</key><string>Parameter Value</string>
 <key>Value</key><string>-1</string>
 <key>Updatable</key><string>No</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Value Flags</key>
 <dict>
 <key>Display Name</key><string>Value Flags</string>
 <key>Value</key><string>NONE</string>
 <key>Updatable</key><string>No</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Deferred Value</key>
 <dict>
 <key>Display Name</key><string>Deferred Value</string>
 <key>Value</key><string>-1</string>
 <key>Updatable</key><string>Yes</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Deferred Value Flags</key>
 <dict>
 <key>Display Name</key><string>Deferred Value Flags</string>
 <key>Value</key><string>INTEGER</string>
 <key>Updatable</key><string>Yes</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Data Type</key>
 <dict>
 <key>Display Name</key><string>Data Type</string>
 <key>Value</key><string>NONE</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Hint</key>
 <string>
 Specifies the priority given to an agent and other database manager
 instance processes and threads by the operating system scheduler.
 Consider rebinding applications after changing this parameter.
 </string>
 </dict>
 <key>Hint</key><string></string>
</dict>
</dict>
<key>Administration</key>
.
.
.
<key>Communication</key>
.
.
.
<key>Diagnostics</key>
.
.
.
<key>Environment</key>
.
.

```

```

.
<key>Miscellaneous</key>
.
.
.
<key>Monitor</key>
.
.
.
<key>Parallel</key>
.
.
.
<key>Performance</key>
.
.
.
</dict>
<key>Database Partition</key>
<dict>
 <key>Display Name</key><string>Database Partition</string>
 <key>0</key>
 <dict>
 <key>Display Name</key><string>0</string>
 <key>Database Configuration Parameter Settings</key>
 <dict>
 <key>Display Name</key>
 <string>Database Configuration Parameter Settings</string>
 <key>Application</key>
 .
 .
 .
 <key>Environment</key>
 <dict>
 <key>Display Name</key><string>Environment</string>
 <key>alt_collate</key>
 <dict>
 <key>Display Name</key><string>alt_collate</string>
 <key>Parameter Value</key>
 <dict>
 <key>Display Name</key><string>Parameter Value</string>
 <key>Value</key><string></string>
 <key>Updatable</key><string>No</string>
 <key>Hint</key><string></string>
 </dict>
 </dict>
 <key>Value Flags</key>
 <dict>
 <key>Display Name</key><string>Value Flags</string>
 <key>Value</key><string>NONE</string>
 <key>Updatable</key><string>No</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Deferred Value</key>
 <dict>
 <key>Display Name</key><string>Deferred Value</string>
 <key>Value</key><string></string>
 <key>Updatable</key><string>Yes</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Deferred Value Flags</key>
 <dict>
 <key>Display Name</key><string>Deferred Value Flags</string>
 <key>Value</key><string>INTEGER</string>
 <key>Updatable</key><string>Yes</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Data Type</key>

```

```

 <dict>
 <key>Display Name</key><string>Data Type</string>
 <key>Value</key><string>NONE</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Hint</key>
 <string>
 Specifies the collating sequence to be used for Unicode tables in a
 non-Unicode database. Until this parameter is set, Unicode tables and
 routines cannot be created in a non-Unicode database. When set, this
 parameter cannot be changed or reset. Default [range] :
 Null [IDENTITY_16BIT].
 </string>
 </dict>
 .
 .
 .
</dict>
<key>Logs</key>
.
.
.
<key>Maintenance</key>
.
.
.
<key>Performance</key>
.
.
.
<key>Recovery</key>
.
.
.
<key>Status</key>
.
.
.
</dict>
<key>Registry Variables Settings</key>
<dict>
 <key>Display Name</key><string>Registry Variables Settings</string>
 <key>DB2CODEPAGE</key>
 <dict>
 <key>Display Name</key><string>DB2CODEPAGE</string>
 <key>Parameter Value</key>
 <dict>
 <key>Display Name</key><string>Parameter Value</string>
 <key>Value</key><string>1208</string>
 <key>Hint</key><string></string>
 </dict>
 </dict>
 <key>Is Aggregate</key>
 <dict>
 <key>Display Name</key><string>Is Aggregate</string>
 <key>Value</key><integer>0</integer>
 <key>Hint</key><string></string>
 </dict>
 <key>Aggregate Name</key>
 <dict>
 <key>Display Name</key><string>Aggregate Name</string>
 <key>Value</key><string></string>
 <key>Hint</key><string></string>
 </dict>
 <key>Level</key>
 <dict>
 <key>Display Name</key><string>Level</string>
 <key>Value</key><string>I</string>

```

```

 <key>Hint</key><string></string>
 </dict>
 <key>Hint</key><string></string>
 </dict>
 .
 .
 .
 </dict>
 <key>Hint</key><string></string>
</dict>
</plist>

```

*Example 3:* Return database and database manager configuration parameters.

```
db2 "call sysproc.get_config(1,0,'en_US',null, null, ?,?)"
```

The following is an example of output from this query:

Value of output parameters

-----

Parameter Name : MAJOR\_VERSION

Parameter Value : 1

Parameter Name : MINOR\_VERSION

Parameter Value : 0

Parameter Name : XML\_OUTPUT

Parameter Value : x'3C3F78.....'

Parameter Name : XML\_MESSAGE

Parameter Value : -

Return Status = 0

The XML output document contains content that is similar to example 2, but does not group the configuration parameters into categories.

*Example 4:* Call the procedure from a function.

```

EXEC SQL BEGIN DECLARE SECTION;
 sqlint16 getconfigMaj;
 sqlint16 getconfigMin;

 SQL TYPE IS BLOB(2M) xmlOutput;
 SQL TYPE IS BLOB(2K) xmlOutMessage;
EXEC SQL END DECLARE SECTION;
 getconfigMaj = 2;
 getconfigMin = 0;

 EXEC SQL CALL SYSPROC.GET_CONFIG(
 :getconfigMaj,
 :getconfigMin,
 'en_US',
 null,
 null,
 :xmlOutput,
 :xmlOutMessage);

```

## GET\_MESSAGE procedure - Get message text

The GET\_MESSAGE procedure returns the short message text, long message text, and SQLSTATE for an SQLCODE.

## Syntax

```
► GET_MESSAGE—(—major_version—,—minor_version—,—requested_locale—,—
► xml_input—,—xml_filter—,—xml_output—,—xml_message—)————►
```

The schema is SYSPROC.

## Procedure parameters

### **major\_version**

An input and output argument of type INTEGER that indicates the major document version. On input, this argument indicates the major document version that the caller supports for the XML documents passed as parameters in the procedure (see the parameter descriptions for *xml\_input*, *xml\_output*, and *xml\_message*). The procedure processes all XML documents in the specified version, or returns an error (+20458) if the version is not valid. On output, this parameter specifies the highest major document version that is supported by the procedure. To determine the highest supported document version, specify NULL for this input parameter and all other required parameters.

If the XML document in the *xml\_input* parameter specifies a Document Type Major Version key, the value for that key must be equal to the value provided in the *major\_version* parameter, or an error (+20458) is raised.

**Supported versions:** 1 and 2

### **minor\_version**

An input and output argument of type INTEGER that indicates the minor document version. On input, this argument specifies the minor document version that the caller supports for the XML documents passed as parameters for this procedure (see the parameter descriptions for *xml\_input*, *xml\_output*, and *xml\_message*). The procedure processes all XML documents in the specified version, or returns an error if the version is not valid. On output, this parameter indicates the highest minor document version that is supported for the highest supported major version. To determine the highest supported document version, specify NULL for this input parameter and all other required parameters.

If the XML document in the *xml\_input* parameter specifies a Document Type Minor Version key, the value for that key must be equal to the value provided in the *minor\_version* parameter, or an error (+20458) is raised.

**Supported versions:** 0

### **requested\_locale**

An input argument of type VARCHAR(33) that specifies a locale. If the specified language is supported on the server, translated content is returned in the *xml\_output* and *xml\_message* parameters. Otherwise, content is returned in the default language. Only the language and possibly the territory information is used from the locale. The locale is not used to format numbers or influence the document encoding. For example, key names and values are not translated. The only translated portion of the XML output and XML message documents are the text for hint, display name, and display unit of each entry. The caller should always compare the requested language to the language that is used in the XML output document (see the document locale entry in the XML output document).

Currently, the only supported value for *requested\_locale* is en\_US.

#### ***xml\_input***

An input argument of type BLOB(32MB) that specifies an XML input document (encoded in UTF-8) that contains input values for the procedure.

For this procedure, the XML input document contains an SQLCODE and uses the following format:

```
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>Document Type Name</key><string>Data Server Message Input</string>
 <key>Required Parameters</key>
 <!-- Specify either SQLCODE or message identifier and message tokens
for the key values below. -->
 <dict>
 <key>SQL Code</key><integer></integer>
 <key>Message Identifier</key><integer></integer>
 <key>Message Tokens</key><array><string>...</string></array>
 </dict>
 <key>Optional Parameters</key>
 <dict>
 <key>Message Token Delimiter<key><string>;</string>
 </key></key></dict>
</dict>
</plist>
```

#### ***xml\_filter***

An input argument of type BLOB(4K) that specifies a valid XPath query string. Use a filter when you want to retrieve a single value from an XML output document. For more information, see the topic that describes XPath filtering.

The following example selects the value for the SQLSTATE from the XML output document: /plist/dict/key[.="SQLSTATE"]/following-sibling::dict[1]/key[.="Value"]/following-sibling::string[1]. If the key is not followed by the specified sibling, an error is returned.

#### ***xml\_output***

An output parameter of type BLOB(32MB) that returns a complete XML output document in UTF-8. If a filter is specified, this parameter returns a string value. If the stored procedure is unable to return a complete output document (for example, if a processing error occurs that results in an SQL warning or error), this parameter is set to NULL.

The XML output is determined by the values that you specify for *major\_version* and *minor\_version*:

Major version	Minor version	xml_output value
NULL	NULL	NULL
1	0	Returns the short text message and SQLSTATE for the corresponding SQLCODE passed in <i>xml_input</i> .
2	0	Returns the short text message, long text message and SQLSTATE for the corresponding SQLCODE passed in <i>xml_input</i> .

When the procedure operates in *complete mode*, this parameter returns an XML document that you can modify and pass back to the procedure as the

*xml\_input* parameter. This approach provides a programmatic way to create valid XML input documents. For more information, see the topic about complete mode.

#### ***xml\_message***

An output parameter of type BLOB(64K) that returns a complete XML output document of type Data Server Message in UTF-8 that provides detailed information about a SQL warning condition. This document is returned when a call to the procedure results in a SQL warning, and the warning message indicates that additional information is returned in the XML message output document. If the warning message does not indicate that additional information is returned, then this parameter is set to NULL.

### **Authorization**

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### **Default PUBLIC privilege**

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

### **Example**

*Example 1:* Return the highest supported version of the procedure.

```
db2 "call sysproc.get_message(null,null,null,null,null,?,?)"
```

The following is an example of output from this query:

Value of output parameters

-----

Parameter Name : MAJOR\_VERSION

Parameter Value : 2

Parameter Name : MINOR\_VERSION

Parameter Value : 0

Parameter Name : XML\_OUTPUT

Parameter Value : -

Parameter Name : XML\_MESSAGE

Parameter Value : -

Return Status = 0

*Example 2:* Run a script called getmsglong.sql to return the short text message and long text message for SQL1034.

getmsglong.sql:

```
call sysproc.get_message(2,0, 'en_US', blob('
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>Document Type Name</key><string>Data Server Message Input</string>
 <key>Document Type Major Version</key><integer>2</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
```

```

 <key>Required Parameters</key>
 <dict>
 <key>SQLCODE</key><string>SQL1034</string>
 </dict>
 </dict>
</plist>'), null, ? , ?)@

```

The following is an example of output from this query:

Value of output parameters

```

Parameter Name : MAJOR_VERSION
Parameter Value : 2

Parameter Name : MINOR_VERSION
Parameter Value : 0

Parameter Name : XML_OUTPUT
Parameter Value : x'3C3F786D6C20766572.....'

Parameter Name : XML_MESSAGE
Parameter Value : -

Return Status = 0

```

The output XML document contains the following content:

```

<plist version="1.0">
<dict>
 <key>Document Type Name</key>
 <string>Data Server Message Output</string>
 <key>Document Type Major Version</key>
 <integer>2</integer>
 <key>Document Type Minor Version</key>
 <integer>0</integer>
 <key>Data Server Product Name</key>
 <string>QDB2/AIX64</string>
 <key>Data Server Product Version</key>
 <string>9.7.0.0</string>
 <key>Data Server Major Version</key>
 <integer>9</integer>
 <key>Data Server Minor Version</key>
 <integer>7</integer>
 <key>Data Server Platform</key>
 <string>AIX 64BIT</string>
 <key>Document Locale</key>
 <string>en_US</string>
 <key>Short Message Text</key>
 <dict>
 <key>Display Name</key><string>Short Message Text</string>
 <key>Value</key>
 <string>
 SQL1034C The database is damaged. All applications processing the database
 have been stopped.
 </string>
 <key>Hint</key><string></string>
 </dict>
 <key>SQLSTATE</key>
 <dict>
 <key>Display Name</key><string>SQLSTATE</string>
 <key>Value</key><string> 58031</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Long Message Text</key>
 <dict>
 <key>Display Name</key><string>Long Message Text</string>
 <key>Value</key>
 <array>

```

```

<string>
SQL1034C The database is damaged. All applications
processing the
</string>
<string> database have been stopped.</string>
<string></string>
<string>Explanation: </string>
<string></string>
<string>
Damage has occurred to the database. It cannot be used until it is
</string>
<string>
recovered. All applications connected to the database have been
</string>
<string>
disconnected and all processes running applications on the
database have
</string>
<string>been stopped.</string>
<string></string>
<string>The command cannot be processed.</string>
<string></string>
<string>User response: </string>
<string></string>
<string>
Issue a RESTART DATABASE command to recover the database. If the RESTART
</string>
<string>
command consistently fails, you may want to restore the database from a
</string>
<string>
backup. In a partitioned database server environment, check the syslog
</string>
<string>
to find out if the RESTART command fails because of node or
</string>
<string>
communication failures before restoring the database from a backup. If
</string>
<string>
so, ensure the database manager is up and running and communication is
</string>
<string>
available among all the nodes, then resubmit the restart command.
</string>
<string></string>
<string>
If you encountered this error during roll-forward processing, you must
</string>
<string>
restore the database from a backup and perform roll-forward again.
</string>
<string></string>
<string>
Note that in a partitioned database environment, the RESTART database
</string>
<string>
command is run on a per-node basis. To ensure that the database is
</string>
<string>restarted on all nodes, use the command: </string>
<string></string>
<string>db2_all db2 restart database</string>
<string><database_name></string>
<string></string>
<string>
This command may have to be run several times to ensure that all
</string>

```

```

 <string>in-doubt transactions have been resolved.</string>
 <string></string>
 <string>
 If you are installing the sample database, drop it and install the
 </string>
 <string>sample database again.</string>
 <string></string>
 <string> sqlcode: -1034</string>
 <string></string>
 <string> sqlstate: 58031</string>
 <string></string>
 <string></string>
 <string></string>
 </array>
 <key>Hint</key><string></string>
</dict>
</dict>
</plist>

```

*Example 3:* Run a script called `getmsgshort.sql` to return only the short text message for SQL1034.

`getmsgshort.sql`:

```

call sysproc.get_message(1,0,'en_US', blob('
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>Document Type Name</key><string>Data Server Message Input</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Required Parameters</key>
 <dict>
 <key>SQLCODE</key><string>SQL1034</string>
 </dict>
</dict>
</plist>'), null, ? , ?)@

```

The following is an example of output from this query:

Value of output parameters

-----

Parameter Name : MAJOR\_VERSION  
Parameter Value : 2

Parameter Name : MINOR\_VERSION  
Parameter Value : 0

Parameter Name : XML\_OUTPUT  
Parameter Value : x'3C3F786D6C20766572.....'

Parameter Name : XML\_MESSAGE  
Parameter Value : -

Return Status = 0

SQL20460W The procedure "SYSPROC.GET\_MESSAGE" supports a higher version, "2", than the specified version, "1", for parameter "1".

The XML output document contains the following content:

```

<plist version="1.0">
<dict><key>Document Type Name</key><string>Data Server Message Output</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Data Server Product Name</key><string>QDB2/AIX64</string>
 <key>Data Server Product Version</key><string>9.7.0.0</string>
 <key>Data Server Major Version</key><integer>9</integer>

```

```

<key>Data Server Minor Version</key><integer>7</integer>
<key>Data Server Platform</key><string>AIX 64BIT</string>
<key>Document Locale</key><string>en_US</string>
<key>Short Message Text</key>
<dict>
 <key>Display Name</key><string>Short Message Text</string>
 <key>Value</key>
 <string>
 SQL1034C The database is damaged. All applications processing the database
 have been stopped.
 </string>
 <key>Hint</key><string></string>
</dict>
<key>SQLSTATE</key>
<dict>
 <key>Display Name</key><string>SQLSTATE</string>
 <key>Value</key><string> 58031</string>
 <key>Hint</key><string></string>
</dict>
</dict>
</plist>

```

*Example 4:* Specify a filter to return the SQLSTATE for SQL1034.

```

db2 "call sysproc.get_message(2,0, 'en_US', blob('
<plist version="1.0">
<dict>
 <key>Document Type Name</key>
 <string>Data Server Message Input</string>
 <key>Required Parameters</key>
 <dict>
 <key>SQLCODE</key><string>SQL1034</string>
 </dict>
</dict>
</plist>'),
blob('/plist/dict/key[.="SQLSTATE"]/following-sibling::dict[1]/
key[.="Value"]/following-sibling::string[1]'), ? , ?)"

```

The following is an example of output from this query:

```

Value of output parameters

Parameter Name : MAJOR_VERSION
Parameter Value : 2

Parameter Name : MINOR_VERSION
Parameter Value : 0

Parameter Name : XML_OUTPUT
Parameter Value : x'203538303331'

Parameter Name : XML_MESSAGE
Parameter Value : -

Return Status = 0

```

The following value is returned for *xml\_output*:

```
58031
```

*Example 5:* Call the procedure from a function.

```

EXEC SQL BEGIN DECLARE SECTION;
 sqlint16 getMsgMaj;
 sqlint16 getMsgMin;

SQL TYPE IS BLOB(2M) xmlOutput;
SQL TYPE IS BLOB(2K) xmlOutMessage;

```

```

EXEC SQL END DECLARE SECTION;
 getMsgMaj = 2;
 getMsgMin = 0;

EXEC SQL CALL SYSPROC.GET_MESSAGE(
 :getMsgMaj,
 :getMsgMin,
 'en_US',
 BLOB('
 <?xml version="1.0" encoding="UTF-8"?>
 <plist version="1.0">
 <dict>
 <key>Document Type Name</key>
 <string>
 Data Server Message Input
 </string>
 <key>Document Type Major Version</key><integer>2</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Required Parameters</key>
 <dict>
 <key>SQLCODE</key><string>SQL1034</string>
 </dict>
 </dict>
 </plist>'),
 null,
 :xmlOutput,
 :xmlOutMessage);

```

## GET\_SYSTEM\_INFO procedure - Get system information

The GET\_SYSTEM\_INFO procedure returns information about the data server, including information about the system, the current instance, installed data server products, environment variables, available CPUs, and other system information.

### Syntax

```

➤➤—GET_SYSTEM_INFO—(—major_version—,—minor_version—,—requested_locale—,—————➤➤
➤—xml_input—,—xml_filter—,—xml_output—,—xml_message—)—————➤➤

```

The schema is SYSPROC.

### Procedure parameters

#### *major\_version*

An input and output argument of type INTEGER that indicates the major document version. On input, this argument indicates the major document version that the caller supports for the XML documents passed as parameters in the procedure (see the parameter descriptions for *xml\_input*, *xml\_output*, and *xml\_message*). The procedure processes all XML documents in the specified version, or returns an error (+20458) if the version is not valid. On output, this parameter specifies the highest major document version that is supported by the procedure. To determine the highest supported document version, specify NULL for this input parameter and all other required parameters.

If the XML document in the *xml\_input* parameter specifies a Document Type Major Version key, the value for that key must be equal to the value provided in the *major\_version* parameter, or an error (+20458) is raised.

**Supported versions:** 1

***minor\_version***

An input and output argument of type INTEGER that indicates the minor document version. On input, this argument specifies the minor document version that the caller supports for the XML documents passed as parameters for this procedure (see the parameter descriptions for *xml\_input*, *xml\_output*, and *xml\_message*). The procedure processes all XML documents in the specified version, or returns an error if the version is not valid. On output, this parameter indicates the highest minor document version that is supported for the highest supported major version. To determine the highest supported document version, specify NULL for this input parameter and all other required parameters.

**Supported versions:** 0

***requested\_locale***

An input argument of type VARCHAR(33) that specifies a locale. If the specified language is supported on the server, translated content is returned in the *xml\_output* and *xml\_message* parameters. Otherwise, content is returned in the default language. Only the language and possibly the territory information is used from the locale. The locale is not used to format numbers or influence the document encoding. For example, key names and values are not translated. The only translated portion of the XML output and XML message documents are the text for hint, display name, and display unit of each entry. The caller should always compare the requested language to the language that is used in the XML output document (see the document locale entry in the XML output document).

Currently, the only supported value for *requested\_locale* is en\_US.

***xml\_input***

Currently, this procedure accepts no input. You must specify NULL for this parameter, or an error (+20458) is raised to indicate that the input is not valid.

***xml\_filter***

An input argument of type BLOB(4K) that specifies a valid XPath query string. Use a filter when you want to retrieve a single value from an XML output document. For more information, see the topic that describes XPath filtering.

The following example selects the value for the Data Server Product Version from the XML output document: `/plist/dict/key[.='Data Server Product Version']/following-sibling::string`. If the key is not followed by the specified sibling, an error is returned.

***xml\_output***

An output parameter of type BLOB(32MB) that returns a complete XML output document in UTF-8. If a filter is specified, this parameter returns a string value. If the stored procedure is unable to return a complete output document (for example, if a processing error occurs that results in an SQL warning or error), this parameter is set to NULL.

The XML output document contains instance information, including information about the fix pack level, release, system information, and environment variables.

***xml\_message***

An output parameter of type BLOB(64K) that returns a complete XML output document of type Data Server Message in UTF-8 that provides detailed information about a SQL warning condition. This document is returned when a call to the procedure results in a SQL warning, and the warning message indicates that additional information is returned in the XML message output

document. If the warning message does not indicate that additional information is returned, then this parameter is set to NULL.

## Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

## Example

*Example 1:* Return the highest version of the procedure.

```
db2 "call sysproc.get_system_info(null,null,null,null,null,?,?)"
```

The following is an example of output from this query:

```
Value of output parameters

Parameter Name : MAJOR_VERSION
Parameter Value : 1

Parameter Name : MINOR_VERSION
Parameter Value : 0

Parameter Name : XML_OUTPUT
Parameter Value : -

Parameter Name : XML_MESSAGE
Parameter Value : -

Return Status = 0
```

*Example 2:* Return system information.

```
db2 "call sysproc.get_system_info(1,0,'en_US',null,null,?,?)"
```

The following is an example of output from this query:

```
Value of output parameters

Parameter Name : MAJOR_VERSION
Parameter Value : 1

Parameter Name : MINOR_VERSION
Parameter Value : 0

Parameter Name : XML_OUTPUT
Parameter Value : x'3C3F786D6C20766572.....

Parameter Name : XML_MESSAGE
Parameter Value : -

Return Status = 0
```

The XML output document contains something similar to the following content:

```

<plist version="1.0">
<dict><key>Document Type Name</key><string>Data Server System Output</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Data Server Product Name</key><string>QDB2/AIX64</string>
 <key>Data Server Product Version</key><string>9.7.0.0</string>
 <key>Data Server Major Version</key><integer>9</integer>
 <key>Data Server Minor Version</key><integer>7</integer>
 <key>Data Server Platform</key><string>AIX 64BIT</string>
 <key>Document Locale</key><string>en_US</string>
 <key>Instance Information</key>
 <dict>
 <key>Display Name</key><string>Instance Information</string>
 <key>Instance Name</key>
 <dict>
 <key>Display Name</key><string>Instance Name</string>
 <key>Value</key><string>myinstance</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Partitionable State</key>
 <dict>
 <key>Display Name</key><string>Partitionable State</string>
 <key>Value</key><integer>0</integer>
 <key>Hint</key><string></string>
 </dict>
 <key>Number of Database Partitions</key>
 <dict>
 <key>Display Name</key><string>Number of Database Partitions</string>
 <key>Value</key><integer>1</integer>
 <key>Hint</key><string></string>
 </dict>
 .
 .
 .
 </dict>
<key>Product Information</key>
<dict>
 <key>Display Name</key><string>Product Information</string>
 .
 .
 .
 <key>DB2_ENTERPRISE_SERVER_EDITION</key>
 <dict>
 <key>Display Name</key><string>DB2_ENTERPRISE_SERVER_EDITION</string>
 <key>Product short name</key>
 <dict>
 <key>Display Name</key><string>Product short name</string>
 <key>Value</key><string>ESE</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Licence</key>
 <dict>
 <key>Display Name</key><string>Licence</string>
 <key>Value</key><string>Y</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Product Release</key>
 <dict>
 <key>Display Name</key><string>Product Release</string>
 <key>Value</key><string>9.7</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Licence type</key>
 <dict>
 <key>Display Name</key><string>Licence type</string>
 <key>Value</key><string>DEVELOPER</string>
 <key>Hint</key><string></string>
 </dict>
 </dict>
</plist>

```

```

 </dict>
 <key>Hint</key><string></string>
 </dict>
 .
 .
 .
<key>Operating System Information</key>
<dict>
 <key>Display Name</key><string>Operating System Information</string>
 <key>Name</key>
 <dict>
 <key>Display Name</key><string>Name</string>
 <key>Value</key><string>AIX</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Version</key>
 <dict>
 <key>Display Name</key><string>Version</string>
 <key>Value</key><string>5</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Release</key>
 <dict>
 <key>Display Name</key><string>Release</string>
 <key>Value</key><string>3</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Hostname</key>
 <dict>
 <key>Display Name</key><string>Hostname</string>
 <key>Value</key><string>achilles</string>
 <key>Hint</key><string></string>
 </dict>
 .
 .
 .
</dict>
<key>Workload Management Configuration</key>
<dict>
 <key>Display Name</key><string>Workload Management Configuration</string>
 <key>Service Class Information</key>
 <dict>
 <key>Display Name</key><string>Service Class Information</string>
 <key>1</key>
 <dict>
 <key>Display Name</key><string>1</string>
 <key>Service Class Name</key>
 <dict>
 <key>Display Name</key><string>Service Class Name</string>
 <key>Value</key><string>SYSDEFAULTSYSTEMCLASS</string>
 <key>Hint</key><string></string>
 </dict>
 </dict>
 <key>Parent Identifier</key>
 <dict>
 <key>Display Name</key><string>Parent Identifier</string>
 <key>Value</key><integer>0</integer>
 <key>Hint</key><string></string>
 </dict>
 <key>Parent Class Name</key>
 <dict>
 <key>Display Name</key><string>Parent Class Name</string>
 <key>Value</key><string></string>
 <key>Hint</key><string></string>
 </dict>
 <key>Creation Time</key>
 <dict>
 <key>Display Name</key><string>Creation Time</string>

```

```

 <key>Value</key><string>2008-04-21-15.14.32.956930</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Alter Time</key>
 <dict>
 <key>Display Name</key><string>Alter Time</string>
 <key>Value</key><string>2008-04-21-15.14.32.956930</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Enabled</key>
 <dict>
 <key>Display Name</key><string>Enabled</string>
 <key>Value</key><string>Y</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Agent Priority</key>
 <dict>
 <key>Display Name</key><string>Agent Priority</string>
 <key>Value</key><integer>-32768</integer>
 <key>Hint</key><string></string>
 </dict>
 <key>Prefetcher Priority</key>
 <dict>
 <key>Display Name</key><string>Prefetcher Priority</string>
 <key>Value</key><string> </string>
 <key>Hint</key><string></string>
 </dict>
 .
 .
 .
</dict>
.
.
.
<key>Workload Information</key>
<dict>
 <key>Display Name</key><string>Workload Information</string>
 <key>1</key>
 <dict>
 <key>Display Name</key><string>1</string>
 <key>Workload Name</key>
 <dict>
 <key>Display Name</key><string>Workload Name</string>
 <key>Value</key><string>SYSDEFAULTUSERWORKLOAD</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Evaluation Order</key>
 <dict>
 <key>Display Name</key><string>Evaluation Order</string>
 <key>Value</key><integer>1</integer>
 <key>Hint</key><string></string>
 </dict>
 <key>Creation Time</key>
 <dict>
 <key>Display Name</key><string>Creation Time</string>
 <key>Value</key><string>2008-04-21-15.14.32.955296</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Alter Time</key>
 <dict>
 <key>Display Name</key><string>Alter Time</string>
 <key>Value</key><string>2008-04-21-15.14.32.955296</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Enabled</key>
 </dict>

```

```

 <key>Display Name</key><string>Enabled</string>
 <key>Value</key><string>Y</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Allow Access</key>
 <dict>
 <key>Display Name</key><string>Allow Access</string>
 <key>Value</key><string>Y</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Service Class Name</key>
 <dict>
 <key>Display Name</key><string>Service Class Name</string>
 <key>Value</key><string>SYSDEFAULTSUBCLASS</string>
 <key>Hint</key><string></string>
 </dict>
 <key>Parent Service Class Name</key>
 <dict>
 <key>Display Name</key><string>Parent Service Class Name</string>
 <key>Value</key><string>SYSDEFAULTUSERCLASS</string>
 <key>Hint</key><string></string>
 </dict>
 .
 .
 .
 </dict>
 <key>Hint</key><string></string>
</dict>
</dict>
</dict></dict></dict></plist>

```

*Example 3:* Call the GET\_SYSTEM\_INFO procure and pass in an unsupported locale.

```
db2 "call sysproc. get_system_info(1,0,'ja_JP',null,null,?,?)"
```

The following is an example of output from this query:

```

Value of output parameters

Parameter Name : MAJOR_VERSION
Parameter Value : 1

Parameter Name : MINOR_VERSION
Parameter Value : 0

Parameter Name : XML_OUTPUT
Parameter Value : x'3C3F786D6C20766572.....

Parameter Name : XML_MESSAGE
Parameter Value : -

Return Status = 0

```

SQL20461W The procedure "SYSPROC.GET\_SYSTEM\_INFO" returned output in the alternate locale, "en\_US", instead of the locale, "ja\_JP", specified in parameter "3". SQLSTATE=01H57

The XML output document will contain the same content that is shown for Example 2.

*Example 4:* Call the procedure from a function.

```

EXEC SQL BEGIN DECLARE SECTION;
sqlint16 getSysInfMaj;
sqlint16 getSysInfMin;

```

```

SQL TYPE IS BLOB(2M) xmlOutput;
SQL TYPE IS BLOB(2K) xmlOutMessage;
EXEC SQL END DECLARE SECTION;
 getSysInfMaj = 1;
 getSysInfMin = 0;

EXEC SQL CALL SYSPROC.GET_SYSTEM_INFO(
 :getSysInfMaj,
 :getSysInfMin,
 'en_US',
 null,
 null,
 :xmlOutput,
 :xmlOutMessage);

```

## SET\_CONFIG procedure - Set configuration parameters

The SET\_CONFIG stored procedure updates the database and database manager configuration parameters that are returned by the GET\_CONFIG procedure.

The SET\_CONFIG procedure accepts an input XML document that contains configuration parameters and their values, uses this information to update the specified configuration parameters, and returns an output XML document that indicates the update status of each configuration parameter.

### Syntax

```

►►—SET_CONFIG—(—major_version—,—minor_version—,—requested_locale—,——————►
►—xml_input—,—xml_filter—,—xml_output—,—xml_message—)—————►

```

The schema is SYSPROC.

### Procedure parameters

#### *major\_version*

An input and output argument of type INTEGER that indicates the major document version. On input, this argument indicates the major document version that the caller supports for the XML documents passed as parameters in the procedure (see the parameter descriptions for *xml\_input*, *xml\_output*, and *xml\_message*). The procedure processes all XML documents in the specified version, or returns an error (+20458) if the version is not valid. On output, this parameter specifies the highest major document version that is supported by the procedure. To determine the highest supported document version, specify NULL for this input parameter and all other required parameters.

If the XML document in the *xml\_input* parameter specifies a Document Type Major Version key, the value for that key must be equal to the value provided in the *major\_version* parameter, or an error (+20458) is raised.

#### Supported versions: 1

#### *minor\_version*

An input and output argument of type INTEGER that indicates the minor document version. On input, this argument specifies the minor document version that the caller supports for the XML documents passed as parameters for this procedure (see the parameter descriptions for *xml\_input*, *xml\_output*, and *xml\_message*). The procedure processes all XML documents in the specified version, or returns an error if the version is not valid. On output, this parameter indicates the highest minor document version that is supported for

the highest supported major version. To determine the highest supported document version, specify NULL for this input parameter and all other required parameters.

If the XML document in the *xml\_input* parameter specifies a Document Type Minor Version key, the value for that key must be equal to the value provided in the *minor\_version* parameter, or an error (+20458) is raised.

**Supported versions:** 0

***requested\_locale***

An input argument of type VARCHAR(33) that specifies a locale. If the specified language is supported on the server, translated content is returned in the *xml\_output* and *xml\_message* parameters. Otherwise, content is returned in the default language. Only the language and possibly the territory information is used from the locale. The locale is not used to format numbers or influence the document encoding. For example, key names and values are not translated. The only translated portion of the XML output and XML message documents are the text for hint, display name, and display unit of each entry. The caller should always compare the requested language to the language that is used in the XML output document (see the document locale entry in the XML output document).

Currently, the only supported value for *requested\_locale* is en\_US.

***xml\_input***

An input argument of type BLOB(32MB) that specifies an XML input document (encoded in UTF-8) that contains input values for the procedure.

For this procedure, the XML input document contains database and database manager configuration settings.

***xml\_filter***

An input argument of type BLOB(4K) that specifies a valid XPath query string. Use a filter when you want to retrieve a single value from an XML output document. For more information, see the topic that describes XPath filtering.

The following example selects the value for a specific configuration parameter setting from the XML output document: `/plist/dict/key[.="Database Manager Configuration Parameter Settings"]/following-sibling::dict[1]/key[3]/following-sibling::dict[1]/dict[1]/key[.="Value"]/following-sibling::string[1]`. If the key is not followed by the specified sibling, an error is returned.

***xml\_output***

An output parameter of type BLOB(32MB) that returns a complete XML output document in UTF-8. If a filter is specified, this parameter returns a string value. If the stored procedure is unable to return a complete output document (for example, if a processing error occurs that results in an SQL warning or error), this parameter is set to NULL.

When this procedure operates in *complete* mode, this parameter returns an XML document that contains the current configuration values set in the server. You can modify this document and pass it back to the procedure as the *xml\_input* parameter. This approach provides a programmatic way to create valid XML input documents.

***xml\_message***

An output parameter of type BLOB(64K) that returns a complete XML output document of type Data Server Message in UTF-8 that provides detailed information about a SQL warning condition. This document is returned when a

call to the procedure results in a SQL warning, and the warning message indicates that additional information is returned in the XML message output document. If the warning message does not indicate that additional information is returned, then this parameter is set to NULL.

## Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

## Example

*Example 1:* Return the latest version of the procedure.

```
db2 "call sysproc.set_config (null,null,null,null,null,?,?)"
```

The following is an example of output from this query:

Value of output parameters

-----

Parameter Name : MAJOR\_VERSION

Parameter Value : 1

Parameter Name : MINOR\_VERSION

Parameter Value : 0

Parameter Name : XML\_OUTPUT

Parameter Value : -

Parameter Name : XML\_MESSAGE

Parameter Value : -

Return Status = 0

*Example 2:* Run a script called setconfig.sql that updates a few database and database manager configuration parameters.

setconfig.sql:

```
call sysproc.set_config(1,0,'en_US',blob('
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>Document Type Name</key><string>Data Server Set Configuration Input</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Document Locale</key><string>en_US</string>
 <key>Database Manager Configuration Parameter Settings</key>
 <dict>
 <key>diaglevel</key><dict><key>Parameter Value</key>
 <dict>
 <key>Value</key><string>4</string>
 </dict>
 </dict>
</dict>
<key>fcm_num_buffers</key>
```

```

<dict>
 <key>Parameter Value</key>
 <dict>
 <key>Value</key><string>4096</string>
 </dict>
 <key>Value Flags</key>
 <dict>
 <key>Value</key><string>MANUAL</string>
 </dict>
</dict>
<key>instance_memory</key>
<dict>
 <key>Deferred Value</key>
 <dict>
 <key>Value</key><string>7424</string>
 </dict>
 <key>Deferred Value Flags</key>
 <dict>
 <key>Value</key><string>AUTOMATIC</string>
 </dict>
</dict>
</dict>
<key>Database Partition</key>
<dict>
 <key>All</key>
 <dict>
 <key>Database Configuration Parameter Settings</key>
 <dict>
 <key>avg_apls</key>
 <dict>
 <key>Parameter Value</key>
 <dict>
 <key>Value</key><string>2</string>
 </dict>
 <key>Value Flags</key>
 <dict>
 <key>Value</key><string>AUTOMATIC</string>
 </dict>
 </dict>
 <key>database_memory</key>
 <dict>
 <key>Deferred Value</key>
 <dict>
 <key>Value</key><string>2</string>
 </dict>
 <key>Deferred Value Flags</key>
 <dict>
 <key>Value</key><string>MANUAL</string>
 </dict>
 </dict>
 </dict>
 </dict>
</dict>
</dict>
</plist>'), null, ?,?)@

```

The following is an example of output from this query:

Value of output parameters

```

Parameter Name : MAJOR_VERSION
Parameter Value : 1

```

```

Parameter Name : MINOR_VERSION
Parameter Value : 0

```

```

Parameter Name : XML_OUTPUT

```

Parameter Value : x'3C3F78...'

Parameter Name : XML\_MESSAGE

Parameter Value : -

Return Status = 0

The output XML document contains something similar to the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<plist version="1.0">
<dict>
 <key>Document Type Name</key>
 <string>Data Server Set Configuration Output</string>
 <key>Document Type Major Version</key><integer>1</integer>
 <key>Document Type Minor Version</key><integer>0</integer>
 <key>Data Server Product Name</key><string>QDB2/AIX64</string>
 <key>Data Server Product Version</key><string>9.7.0.0</string>
 <key>Data Server Major Version</key><integer>9</integer>
 <key>Data Server Minor Version</key><integer>7</integer>
 <key>Data Server Platform</key><string>AIX 64BIT</string>
 <key>Document Locale</key><string>en_US</string>
 <key>Database Manager Configuration Parameter Settings</key>
 <dict>
 <key>Display Name</key>
 <string>Database Manager Configuration Parameter Settings</string>
 <key>diaglevel</key>
 <dict>
 <key>Display Name</key><string>diaglevel</string>
 <key>Parameter Value</key>
 <dict>
 <key>Display Name</key><string>Parameter Value</string>
 <key>Value</key><string>4</string>
 </dict>
 <key>Parameter Update Status</key>
 <dict>
 <key>Display Name</key><string>Parameter Update Status</string>
 <key>SQLCODE</key>
 <dict>
 <key>Display Name</key><string>SQLCODE</string>
 <key>Value</key><integer>0</integer>
 </dict>
 </dict>
 <key>Message Tokens</key>
 <dict>
 <key>Display Name</key><string>Message Tokens</string>
 <key>Value</key><array><string></string></array>
 </dict>
 <key>SQLSTATE</key>
 <dict>
 <key>Display Name</key><string>SQLSTATE</string>
 <key>Value</key><string></string>
 </dict>
 </dict>
 </dict>
 <key>fcm_num_buffers</key>
 <dict>
 <key>Display Name</key><string>fcm_num_buffers</string>
 <key>Parameter Value</key>
 <dict>
 <key>Display Name</key><string>Parameter Value</string>
 <key>Value</key><string>4096</string>
 </dict>
 <key>Value Flags</key>
 <dict>
 <key>Display Name</key><string>Value Flags</string>
 <key>Value</key><string>MANUAL</string>
 </dict>
 </dict>
</plist>
```

```

<key>Parameter Update Status</key>
<dict>
 <key>Display Name</key><string>Parameter Update Status</string>
 <key>SQLCODE</key><dict>
 <key>Display Name</key><string>SQLCODE</string>
 <key>Value</key> <integer>0</integer>
 </dict>
</dict>
<key>Message Tokens</key>
<dict>
 <key>Display Name</key><string>Message Tokens</string>
 <key>Value</key><array><string></string></array>
</dict>
<key>SQLSTATE</key>
<dict>
 <key>Display Name</key><string>SQLSTATE</string>
 <key>Value</key><string></string>
</dict>
</dict>
</dict>
<key>instance_memory</key>
<dict>
 <key>Display Name</key><string>instance_memory</string>
 <key>Deferred Value</key>
 <dict>
 <key>Display Name</key><string>Deferred Value</string>
 <key>Value</key><string>7424</string>
 </dict>
 <key>Deferred Value Flags</key>
 <dict>
 <key>Display Name</key><string>Deferred Value Flags</string>
 <key>Value</key><string>AUTOMATIC</string>
 </dict>
</dict>
<key>Parameter Update Status</key>
<dict>
 <key>Display Name</key><string>Parameter Update Status</string>
 <key>SQLCODE</key>
 <dict>
 <key>Display Name</key><string>SQLCODE</string>
 <key>Value</key><integer>0</integer>
 </dict>
 <key>Message Tokens</key>
 <dict>
 <key>Display Name</key><string>Message Tokens</string>
 <key>Value</key><array><string></string></array>
 </dict>
 <key>SQLSTATE</key>
 <dict>
 <key>Display Name</key><string>SQLSTATE</string>
 <key>Value</key><string></string>
 </dict>
</dict>
</dict>
</dict>
<key>Database Partition</key>
<dict>
 <key>Display Name</key><string>Database Partition</string>
 <key>All</key>
 <dict>
 <key>Display Name</key><string>All</string>
 </dict>
 <key>Database Configuration Parameter Settings</key>
 <dict>
 <key>Display Name</key>
 <string>Database Configuration Parameter Settings</string>
 <key>avg_appls</key>
 <dict>
 <key>Display Name</key><string>avg_appls</string>
 <key>Parameter Value</key>
 </dict>
 </dict>
</dict>

```

```

 <dict>
 <key>Display Name</key><string>Parameter Value</string>
 <key>Value</key><string>2</string>
 </dict>
 <key>Value Flags</key>
 <dict>
 <key>Display Name</key><string>Value Flags</string>
 <key>Value</key><string>AUTOMATIC</string>
 </dict>
 <key>Parameter Update Status</key>
 <dict>
 <key>Display Name</key><string>Parameter Update Status</string>
 <key>Update Coverage</key>
 <dict>
 <key>Display Name</key><string>Update Coverage</string>
 <key>Value</key><string>Complete</string>
 </dict>
 <key>SQLCODE</key>
 <dict>
 <key>Display Name</key><string>SQLCODE</string>
 <key>Value</key><integer>0</integer>
 </dict>
 <key>Message Tokens</key>
 <dict>
 <key>Display Name</key><string>Message Tokens</string>
 <key>Value</key><array><string></string> </array>
 </dict>
 <key>SQLSTATE</key>
 <dict>
 <key>Display Name</key><string>SQLSTATE</string>
 <key>Value</key><string></string>
 </dict>
 </dict>
</dict>
<key>database_memory</key>
<dict>
 <key>Display Name</key><string>database_memory</string>
 <key>Deferred Value</key>
 <dict>
 <key>Display Name</key><string>Deferred Value</string>
 <key>Value</key><string>2</string>
 </dict>
 <key>Deferred Value Flags</key>
 <dict>
 <key>Display Name</key><string>Deferred Value Flags</string>
 <key>Value</key><string>MANUAL</string>
 </dict>
 <key>Parameter Update Status</key>
 <dict>
 <key>Display Name</key><string>Parameter Update Status</string>
 <key>Update Coverage</key>
 <dict>
 <key>Display Name</key><string>Update Coverage</string>
 <key>Value</key><string>Complete</string>
 </dict>
 <key>SQLCODE</key>
 <dict>
 <key>Display Name</key><string>SQLCODE</string>
 <key>Value</key><integer>0</integer>
 </dict>
 <key>Message Tokens</key>
 <dict>
 <key>Display Name</key><string>Message Tokens</string>
 <key>Value</key><array><string></string></array>
 </dict>
 <key>SQLSTATE</key>
 </dict>

```





```

<key>Document Type Minor Version</key><integer>0</integer>
<key>Document Locale</key><string>en_US</string>
<key>Database Manager Configuration Parameter Settings</key>
<dict>
 <key>diaglevel</key><dict><key>Parameter Value</key>
 <dict>
 <key>Value</key><string>4</string>
 </dict>
 </dict>
<key>fcm_num_buffers</key>
<dict>
 <key>Parameter Value</key>
 <dict>
 <key>Value</key><string>4096</string>
 </dict>
 <key>Value Flags</key>
 <dict>
 <key>Value</key><string>MANUAL</string>
 </dict>
</dict>
<key>instance_memory</key>
<dict>
 <key>Deferred Value</key>
 <dict>
 <key>Value</key><string>7424</string>
 </dict>
 <key>Deferred Value Flags</key>
 <dict>
 <key>Value</key><string>AUTOMATIC</string>
 </dict>
</dict>
</dict>
<key>Database Partition</key>
<dict>
 <key>All</key>
 <dict>
 <key>Database Configuration Parameter Settings</key>
 <dict>
 <key>avg_appls</key>
 <dict>
 <key>Parameter Value</key>
 <dict>
 <key>Value</key><string>2</string>
 </dict>
 <key>Value Flags</key>
 <dict>
 <key>Value</key><string>AUTOMATIC</string>
 </dict>
 </dict>
 <key>database_memory</key>
 <dict>
 <key>Deferred Value</key>
 <dict>
 <key>Value</key><string>2</string>
 </dict>
 <key>Deferred Value Flags</key>
 <dict>
 <key>Value</key><string>MANUAL</string>
 </dict>
 </dict>
 </dict>
</dict>
</dict>
</dict>
</dict>
</dict>

```

```

 </plist>'),
 null,
 :xmlOutput,
 :xmlOutMessage);

```

## Configuration routines and views

This grouping of routines and views can be used to retrieve database configuration parameter information and database manager configuration parameter information.

### DBCFCG administrative view and DB\_GET\_CFG table function - Retrieve database configuration parameter information

The DBCFCG administrative view and the DB\_GET\_CFG table function retrieve database configuration parameter information for the currently connected database for all active database members.

To ensure that information is retrieved from all database members in the instance, run the **ACTIVATE DATABASE** command before using the administrative view or table function. Depending on whether you are using the administrative view or the table function, refer to one of the following sections:

- “DBCFCG administrative view”
- “DB\_GET\_CFG table function” on page 364

### DBCFCG administrative view

The schema is SYSIBMADM.

#### Authorization

One of the following authorizations is required:

- SELECT privilege on the DBCFCG administrative view
- CONTROL privilege on the DBCFCG administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

In a nonrestrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

### Examples

*Example 1:* Retrieve the automatic maintenance settings in the database configuration that are stored in memory for all active database members.

```
SELECT DBPARTITIONNUM, NAME, VALUE FROM SYSIBMADM.DBCFCG WHERE NAME LIKE 'auto_%'
```

The following is an example of output for this query.

DBPARTITIONNUM	NAME	VALUE
0	auto_maint	OFF
0	auto_db_backup	OFF
0	auto_tbl_maint	OFF
0	auto_runstats	OFF

0 auto_reorg	OFF
0 autorestart	ON

6 record(s) selected.

*Example 2:* Retrieve all the database configuration parameters values stored on disk for all active database members.

```
SELECT NAME, DEFERRED_VALUE, DBPARTITIONNUM FROM SYSIBMADM.DBCFG
```

The following is an example of output for this query.

NAME	DEFERRED_VALUE	DBPARTITIONNUM
app_ctl_heap_sz	128	0
appgroup_mem_sz	30000	0
applheapsz	256	0
archretrydelay	20	0
...		
autorestart	ON	0
avg_appls	1	0
blk_log_dsk_ful	NO	0
catalogcache_sz	-1	0
...		

## DB\_GET\_CFG table function

### Syntax

```
DB_GET_CFG (member)
```

The schema is SYSPROC.

### Table function parameter

#### *member*

An optional input argument of type INTEGER that specifies the number of a database member in the same instance as the currently connected database. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly. On an active database member, the database is available for connection and use by applications.

### Authorization

One of the following authorizations is required:

- EXECUTE privilege on the DB\_GET\_CFG table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

In a nonrestrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Example

In a Db2 pureScale environment, retrieve the automatic maintenance settings in the database configuration that are stored in memory for all active members.

```
SELECT NAME, VARCHAR(VALUE, 20) AS VALUE, MEMBER,
 DBPARTITIONNUM FROM TABLE(SYSPROC.DB_GET_CFG(-2))
WHERE NAME LIKE 'auto_%' ORDER BY NAME, MEMBER
```

The following is an example of output from this query.

NAME	VALUE	MEMBER	DBPARTITIONNUM
auto_db_backup	OFF	0	0
auto_db_backup	OFF	1	0
auto_db_backup	OFF	2	0
auto_del_rec_obj	OFF	0	0
auto_del_rec_obj	OFF	1	0
auto_del_rec_obj	OFF	2	0
auto_maint	ON	0	0
auto_maint	ON	1	0
auto_maint	ON	2	0
auto_reorg	OFF	0	0
auto_reorg	OFF	1	0
auto_reorg	OFF	2	0
auto_reval	DEFERRED	0	0
auto_reval	DEFERRED	1	0
auto_reval	DEFERRED	2	0
auto_runstats	ON	0	0
auto_runstats	ON	1	0
auto_runstats	ON	2	0
auto_stats_views	OFF	0	0
auto_stats_views	OFF	1	0
auto_stats_views	OFF	2	0
auto_stmt_stats	ON	0	0
auto_stmt_stats	ON	1	0
auto_stmt_stats	ON	2	0
auto_tbl_maint	ON	0	0
auto_tbl_maint	ON	1	0
auto_tbl_maint	ON	2	0
autorestart	ON	0	0
autorestart	ON	1	0
autorestart	ON	2	0

30 record(s) selected.

## Information returned

Table 59. Information returned by the DBCFG administrative view and the DB\_GET\_CFG table function

Column name	Data type	Description
NAME	VARCHAR(32)	Configuration parameter name.
VALUE	VARCHAR(1024)	The current value of the configuration parameter stored in memory.

Table 59. Information returned by the DBCFG administrative view and the DB\_GET\_CFG table function (continued)

Column name	Data type	Description
VALUE_FLAGS	VARCHAR(10)	Provides specific information for the configuration parameter current value. Valid values are: <ul style="list-style-type: none"> <li>• NONE - no additional information</li> <li>• AUTOMATIC - the configuration parameter has been set to automatic</li> <li>• COMPUTED - the configuration parameter has been set to a computed value</li> </ul>
DEFERRED_VALUE	VARCHAR(1024)	The value of the configuration parameter on disk. For some database configuration parameters, changes only take effect when the database is reactivated. In these cases, all applications must first disconnect from the database. (If the database was activated, then it must be deactivated and reactivated.) The changes take effect at the next connection to the database.
DEFERRED_VALUE_FLAGS	VARCHAR(10)	Provides specific information for the configuration parameter deferred value. Valid values are: <ul style="list-style-type: none"> <li>• NONE - no additional information</li> <li>• AUTOMATIC - the configuration parameter</li> <li>• COMPUTED - the configuration parameter has been set to a computed value</li> </ul>
DATATYPE	VARCHAR(128)	Configuration parameter data type.
DBPARTITIONNUM	SMALLINT	dbpartitionnum - Database partition number monitor element
MEMBER	SMALLINT	member - Database member monitor element

## DBMCFG administrative view and DBM\_GET\_CFG table function - Retrieve database manager configuration parameter information

The DBMCFG administrative view and the DBM\_GET\_CFG table function returns database manager configuration parameter information including the values in memory and the values stored on disk.

Depending on whether you are using the administrative view or the table function, refer to one of the following sections:

- “DBMCFG administrative view”
- “DBM\_GET\_CFG table function” on page 369

### DBMCFG administrative view

The schema is SYSIBMADM.

#### Authorization

One of the following authorizations is required:

- SELECT privilege on the DBMCFG administrative view
- CONTROL privilege on the DBMCFG administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

#### Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

#### Examples

*Example 1:* Retrieve values for all the database manager configuration parameters stored on disk:

```
SELECT NAME, DEFERRED_VALUE FROM SYSIBMADM.DBMCFG
```

The following is an example of output for this query.

NAME	DEFERRED_VALUE
agent_stack_sz	0
agentpri	-1
alt_diagpath	
alt_diagpath_resolved	
alternate_auth_enc	AES_ONLY
aslheapsz	15
audit_buf_sz	0
authentication	SERVER
catalog_noauth	YES
cf_diaglevel	2
cf_diagpath	/home/hotel1nx93/db2docs2/sql1lib/db2dump/
cf_diagpath_resolved	/home/hotel1nx93/db2docs2/sql1lib/db2dump/
cf_mem_sz	131072
cf_num_conns	16
cf_num_workers	1
clnt_krb_plugin	
...	
comm_bandwidth	0.000000e+00
comm_exit_list	

```

conn_elapse 0
cpuspeed 4.000000e-05
cur_eff_arch_lv1 V:10 R:5 M:0 F:1 I:0 SB:0
cur_eff_code_lv1 V:10 R:5 M:0 F:1 I:0 SB:0
dft_account_str
dft_mon_bufpool OFF
...
dft_mon_timestamp ON
dft_mon_uow OFF
...
jdk_path /home/hotel1nx93/db2docs2/sql1lib/java/jdk64
...
ssl_svcname 22711
ssl_svr_keydb /GSKit/Keystore/key.kdb
ssl_svr_label
ssl_svr_stash /GSKit/Keystore/key.sth

```

*Example 2:* Retrieve all the database manager configuration parameters values.

```
SELECT * FROM SYSIBMADM.DBMCFG
```

The following is an example of output for this query.

NAME	VALUE	VALUE_FLAGS	...
agent_stack_sz	0	NONE	...
agentpri	-1	NONE	...
alt_diagpath		NONE	...
alt_diagpath_resolved		NONE	...
alternate_auth_enc	NOT_SPECIFIED	NONE	...
aslheapsz	15	NONE	...
audit_buf_sz	0	NONE	...
authentication	SERVER	NONE	...
catalog_noauth	YES	NONE	...
cf_diaglevel	2	NONE	...
cf_diagpath	/home/userc/sql1lib/db2dump/	NONE	...
cf_diagpath_resolved	/home/userc/sql1lib/db2dump/	NONE	...
cf_mem_sz	131072	AUTOMATIC	...
cf_num_conns	16	AUTOMATIC	...
cf_num_workers	1	AUTOMATIC	...
clnt_krb_plugin		NONE	...
clnt_pw_plugin		NONE	...
cluster_mgr		NONE	...
comm_bandwidth	0.000000e+00	NONE	...
comm_exit_list		NONE	...
conn_elapse	0	NONE	...
cpuspeed	4.000000e-05	NONE	...
cur_eff_arch_lv1	V:10 R:5 M:0 F:1 I:0 SB:0	NONE	...
cur_eff_code_lv1	V:10 R:5 M:0 F:1 I:0 SB:0	NONE	...
dft_account_str		NONE	...
dft_mon_bufpool	OFF	NONE	...
dft_mon_lock	OFF	NONE	...
dft_mon_sort	OFF	NONE	...
dft_mon_stmt	OFF	NONE	...
dft_mon_table	OFF	NONE	...
dft_mon_timestamp	ON	NONE	...
dft_mon_uow	OFF	NONE	...
dftdbpath	/home/userb	NONE	...
diaglevel	3	NONE	...
diagpath	/home/userc/tmp/ \$m	NONE	...
diagpath_resolved	/home/userc/tmp/DIAG000	NONE	...
diagsize	0	NONE	...
dir_cache	YES	NONE	...
discover	SEARCH	NONE	...
discover_inst	ENABLE	NONE	...

fcm_num_anchors	0	AUTOMATIC	...
fcm_num_buffers	0	AUTOMATIC	...
fcm_num_channels	256	AUTOMATIC	...
...			

Output for this query (continued).

... DEFERRED_VALUE	DEFERRED_VALUE_FLAGS	DATATYPE
...	-----	-----
... 0	NONE	INTEGER
... -1	NONE	INTEGER
...	NONE	VARCHAR(215)
...	NONE	VARCHAR(215)
... AES_ONLY	NONE	VARCHAR(32)
... 15	NONE	BIGINT
... 0	NONE	BIGINT
... SERVER	NONE	VARCHAR(32)
... YES	NONE	VARCHAR(3)
... 2	NONE	INTEGER
... /home/userc/sql1lib/db2dump/	NONE	VARCHAR(215)
... /home/userc/sql1lib/db2dump/	NONE	VARCHAR(215)
... 131072	AUTOMATIC	INTEGER
... 16	AUTOMATIC	INTEGER
... 1	AUTOMATIC	INTEGER
...	NONE	VARCHAR(32)
...	NONE	VARCHAR(32)
...	NONE	VARCHAR(262)
... 0.000000e+00	NONE	REAL
...	NONE	VARCHAR(128)
... 0	NONE	INTEGER
... 4.000000e-05	NONE	REAL
... V:10 R:5 M:0 F:1 I:0 SB:0	NONE	VARCHAR(50)
... V:10 R:5 M:0 F:1 I:0 SB:0	NONE	VARCHAR(50)
...	NONE	VARCHAR(25)
... OFF	NONE	VARCHAR(3)
... OFF	NONE	VARCHAR(3)
... OFF	NONE	VARCHAR(3)
... OFF	NONE	VARCHAR(3)
... OFF	NONE	VARCHAR(3)
... OFF	NONE	VARCHAR(3)
... ON	NONE	VARCHAR(3)
... OFF	NONE	VARCHAR(3)
... /home/userb	NONE	VARCHAR(215)
... 3	NONE	INTEGER
... /home/userc/tmp \$m	NONE	VARCHAR(215)
... /home/userc/tmp/DIAG000	NONE	VARCHAR(215)
... 0	NONE	BIGINT
... YES	NONE	VARCHAR(3)
... SEARCH	NONE	VARCHAR(8)
... ENABLE	NONE	VARCHAR(8)
... 0	AUTOMATIC	BIGINT
... 512	AUTOMATIC	BIGINT
... 256	AUTOMATIC	INTEGER
...		

## DBM\_GET\_CFG table function

### Syntax

►► DBM\_GET\_CFG (—) ◀◀

The schema is SYSPROC.

## Authorization

One of the following authorizations is required:

- EXECUTE privilege on the DBM\_GET\_CFG table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a nonrestrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Example

Retrieve the workload manager settings from the database manager configuration:

```
SELECT NAME, VARCHAR(VALUE, 30) AS VALUE
 FROM TABLE(SYSPROC.DBM_GET_CFG())
 WHERE NAME LIKE 'wlm%'
 ORDER BY NAME
```

The following is an example of output from this query.

NAME	VALUE
wlm_disp_concur	128
wlm_disp_cpu_shares	NO
wlm_disp_min_util	5
wlm_dispatcher	YES

4 record(s) selected.

## Usage Notes

If **instance\_memory** is configured per-member on DPF or Db2 pureScale instances, then the deferred value for individual members is returned as the **member\_inst\_mem** configuration parameter name rather than through **instance\_memory** for individual members. For more details, see [instance\\_memory - Instance memory configuration parameter](#)

## Information returned

Table 60. Information returned by the DBMCFG administrative view and DBM\_GET\_CFG table function

Column name	Data type	Description
NAME	VARCHAR(32)	Configuration parameter name.
VALUE	VARCHAR(256)	The current value of the configuration parameter stored in memory.

Table 60. Information returned by the DBMCFG administrative view and DBM\_GET\_CFG table function (continued)

Column name	Data type	Description
VALUE_FLAGS	VARCHAR(10)	Provides specific information for the configuration parameter current value. Valid values are: <ul style="list-style-type: none"> <li>NONE - no additional information</li> <li>AUTOMATIC - the configuration parameter has been set to automatic</li> </ul>
DEFERRED_VALUE	VARCHAR(256)	The value of the configuration parameter on disk. For some database manager configuration parameters, the database manager must be stopped ( <b>db2stop</b> ) and restarted ( <b>db2start</b> ) for this value to take effect.
DEFERRED_VALUE_FLAGS	VARCHAR(10)	Provides specific information for the configuration parameter deferred value. Valid values are: <ul style="list-style-type: none"> <li>NONE - no additional information</li> <li>AUTOMATIC - the configuration parameter has been set to automatic</li> </ul>
DATATYPE	VARCHAR(128)	Configuration parameter data type.

## Db2 pureScale instance information routines and views

In a Db2 pureScale environment, certain routines provide information about the state of the Db2 pureScale instance and its members.

### DB\_MEMBERS table function

The DB\_MEMBERS table function returns basic member information about a Db2 pureScale instance.

#### Syntax

►►—DB\_MEMBERS—(—)—————►◄

The schema is SYSPROC.

#### Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority

- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

### Table function parameters

The function has no input parameters.

### Example

Retrieve instance information:

```
$ db2 "select * from table(SYSPROC.DB_MEMBERS()) as members"
```

The following is an example of output from this query.

MEMBER_NUMBER	HOST_NAME	PARTITION_NUMBER	MEMBER_TYPE
0	member1.mycompany.com	0	D
1	member2.mycompany.com	0	D
2	member3.mycompany.com	0	D
128	ca1.mycompany.com	0	C
129	ca2.mycompany.com	0	C

PORT_NUMBER	SWITCH_NAME	STATUS
0	-	-
0	-	-
0	-	-
0	-	-
0	-	-

5 record(s) selected.

### Information returned

Table 61. Information returned by the DB\_MEMBERS table function

Column name	Data type	Description
MEMBER_NUMBER	SMALLINT	member - Database member monitor element
HOST_NAME	VARCHAR(256)	host_name - Host name monitor element
PARTITION_NUMBER	SMALLINT	partition_number - Partition Number monitor element
MEMBER_TYPE	CHAR(1)	The type of member: <ul style="list-style-type: none"> <li>• C - a CF member</li> <li>• D - a database member</li> </ul>
PORT_NUMBER	SMALLINT	port_number - Port number
SWITCH_NAME	VARCHAR(128)	The switch name of the member on the host
STATUS	SMALLINT	Reserved for future use

## DB2\_CLUSTER\_HOST\_STATE administrative view and DB2\_GET\_CLUSTER\_HOST\_STATE table function - get information about hosts

The DB2\_CLUSTER\_HOST\_STATE administrative view and DB2\_GET\_CLUSTER\_HOST\_STATE table function retrieve information about the hosts that are part of a Db2 pureScale instance.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “DB2\_CLUSTER\_HOST\_STATE administrative view”
- “DB2\_GET\_CLUSTER\_HOST\_STATE table function” on page 374

### DB2\_CLUSTER\_HOST\_STATE administrative view

The DB2\_CLUSTER\_HOST\_STATE administrative view returns the hosts that are part of a Db2 pureScale instance. You can obtain a list of unique hosts plus their associated state information for the instance.

The schema is SYSIBMADM.

Refer to the Information returned for DB2\_CLUSTER\_HOST\_STATE administrative view and DB2\_GET\_CLUSTER\_HOST\_STATE table for a complete list of information that can be returned.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the DB2\_CLUSTER\_HOST\_STATE administrative view
- CONTROL privilege on the DB2\_CLUSTER\_HOST\_STATE administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Example

The following example uses the DB2\_CLUSTER\_HOST\_STATE administrative view to obtain information for a Db2 pureScale instance with four members and two CFs.

```
SELECT * FROM SYSIBMADM.DB2_CLUSTER_HOST_STATE
```

The output from this query looks like this:

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
so1	ACTIVE	NO	NO
so2	ACTIVE	NO	NO
so3	ACTIVE	NO	NO
so4	ACTIVE	NO	NO
so5	ACTIVE	NO	NO
so6	ACTIVE	NO	NO

6 record(s) selected.

## DB2\_GET\_CLUSTER\_HOST\_STATE table function

The DB2\_GET\_CLUSTER\_HOST\_STATE table function returns the same information as the DB2\_CLUSTER\_HOST\_STATE administrative view, but allows you to specify a host name.

Refer to the Information returned for DB2\_CLUSTER\_HOST\_STATE administrative view and DB2\_GET\_CLUSTER\_HOST\_STATE table for a complete list of information that can be returned.

### Syntax

►►—DB2\_GET\_CLUSTER\_HOST\_STATE—(—*hostname*—)————►►

The schema is SYSPROC.

### Table function parameters

*hostname*

An optional input argument of type VARCHAR(255) that specifies the hostname in short or long form, for which records will be returned. If the IP address is provided instead, no records will be returned. If this parameter is null or an empty string ("), all records are returned.

### Authorization

One of the following authorizations is required:

- EXECUTE privilege on the DB2\_GET\_CLUSTER\_HOST\_STATE table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Example

The following example uses the DB2\_GET\_CLUSTER\_HOST\_STATE table function to retrieve information about the host named so3 in a Db2 pureScale instance with four members and two CFs.

```
SELECT * FROM TABLE(DB2_GET_CLUSTER_HOST_STATE('so3')) as T
```

The following is an example of the output from this query:

HOSTNAME	STATE	INSTANCE_STOPPED	ALERT
so3	ACTIVE	NO	NO

1 record(s) selected.

## Information returned

Table 62. Information returned for DB2\_CLUSTER\_HOST\_STATE administrative view and DB2\_GET\_CLUSTER\_HOST\_STATE

Column name	Data type	Description
HOSTNAME	VARCHAR(255)	hostname - Host name monitor element
STATE	VARCHAR(32)	The hosts state information: <ul style="list-style-type: none"><li>• ACTIVE - host is available for use by the database manager</li><li>• INACTIVE - host is not available for use by the database manager</li><li>• NULL - host state is inapplicable (for example, it is in a partitioned database environment)</li></ul>
INSTANCE_STOPPED	VARCHAR(8)	Whether the instance is stopped or not on the hosts: <ul style="list-style-type: none"><li>• YES - instance is stopped on the host</li><li>• NO - instance is not stopped on the host</li><li>• NULL - inapplicable (for example, if in a partitioned database environment)</li></ul>
ALERT	VARCHAR(8)	Information about alerts on the hosts: <ul style="list-style-type: none"><li>• YES - there is an alert on the host</li><li>• NO - there is no alert on the host</li><li>• NULL - inapplicable (for example, if in a partitioned database environment)</li></ul>

## DB2\_INSTANCE\_ALERTS view - get information about alerts

The DB2\_INSTANCE\_ALERTS administrative view provides information about alerts in the Db2 pureScale instance.

The schema is SYSIBMADM.

## Authorization

The following authorizations is required:

- SELECT privilege on the DB2\_INSTANCE\_ALERTS administrative view

## Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

## Example

The following example demonstrates the kind of information you can retrieve using the DB2\_INSTANCE\_ALERTS administrative view.

```
SELECT * FROM SYSIBMADM.DB2_INSTANCE_ALERTS
```

The following is an example of output for this query. The columns have been placed on separate rows for readability.

MESSAGE

```

Restart light failed for member 0 on host(s) hostC. Check the db2diag.log for
messages concerning a restart light or database crash recovery failure on these
hosts for member 0. See the IBM Knowledge Center for more details.
```

#### ACTION

-----  
This alert must be cleared manually with the command:  
db2cluster -clear -alert -member 0

#### IMPACT

-----  
Restart light will not succeed for member 0 on the hosts listed until this alert has been cleared.

### Information returned

*Table 63. Information returned by the DB2\_INSTANCE\_ALERTS administrative view*

Column name	Data type	Description or Monitor element
MESSAGE	VARCHAR(32672)	message - Control Table Message monitor element
ACTION	VARCHAR(32672)	Action required to clear the alert.
IMPACT	VARCHAR(32672)	The impact to the Db2 pureScale instance if the alert is not cleared.

### DB2\_MEMBER and DB2\_CF administrative views and DB2\_GET\_INSTANCE\_INFO table function

The DB2\_MEMBER and DB2\_CF administrative views and DB2\_GET\_INSTANCE\_INFO table function return information about database members of single and multi-partitioned instances and the members and CFs of a Db2 pureScale instance, including state information where applicable.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “DB2\_MEMBER and DB2\_CF administrative views”
- “DB2\_GET\_INSTANCE\_INFO table function” on page 377

### DB2\_MEMBER and DB2\_CF administrative views

The DB2\_MEMBER administrative view returns such information as the machine name on which a member is currently running, its state, whether any alerts are present, and the name of the high speed interconnect for internal database communications.

The DB2\_CF administrative view returns similar information for cluster caching facilities (also known as CFs).

The schema is SYSIBMADM.

Refer to the tables in “Information returned” on page 379 for a complete list of information that can be returned.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the administrative view

- CONTROL privilege on the administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Examples

The following example uses the DB2\_MEMBER administrative view to display the status of all members in the database instance. This is appropriate for the following kinds of instance:

- A Db2 pureScale instance
- A partitioned or non-partitioned database instance using the High Availability Disaster Recovery (HADR) feature.

Execute the following query to display the status of all members in the database instance:

```
SELECT * FROM SYSIBMADM.DB2_MEMBER
```

The following is an example of output from this query.

ID	HOME_HOST	CURRENT_HOST	STATE	ALERT
0	so1	so1	STARTED	NO
2	so2	so2	STARTED	NO
4	so3	so3	STARTED	NO

3 record(s) selected.

The following example uses the DB2\_CF administrative view to display the status of all CFs in the database instance. This is appropriate for a Db2 pureScale instance, only.

Execute the following query to display the status of all cluster caching facilities in the database instance:

```
SELECT * FROM SYSIBMADM.DB2_CF
```

The following is an example of output from this query.

ID	CURRENT_HOST	STATE	ALERT
128	so5	PRIMARY	NO
129	so6	PEER	NO

2 record(s) selected.

## DB2\_GET\_INSTANCE\_INFO table function

The DB2\_GET\_INSTANCE\_INFO table function returns the same information as the DB2\_MEMBER and DB2\_CF administrative views, but enables you to filter the information returned by passing input parameters, such as the current host. The current host is the host where the member is currently running.

## Syntax

```
►►DB2_GET_INSTANCE_INFO(—id—,—home_host—,—current_host—,—type—,—db_partition_num—)◄◄
```

The schema is SYSPROC.

## Table function parameters

*id* An optional input argument of type INTEGER that specifies a valid member or cluster caching facility identifier. Entries are returned for all database members or cluster caching facilities that match the input. If this parameter is null or -2 all records are returned. If this parameter is -1, information for the currently connected member is returned.

*home\_host*

An optional input argument of type VARCHAR(255) that specifies the *home host* for which records are to be returned. The home host is the original host name associated with a particular member when the member was created. Use the short format of the host name and not an IP address for the *home\_host* argument. If this parameter is null or an empty string, all records are returned.

*current\_host*

An optional input argument of type VARCHAR(255) that specifies the current host for which records are to be returned. The current host is the host where the member is currently running, which might not be the same as the home host, if, for example, the member had to be started on another host as part of a restart light operation. Use the short format of the host name and not an IP address for the *current\_host* argument.

*type*

An optional input argument of type VARCHAR(32) that specifies whether to retrieve information for members or cluster caching facilities. The possible values are:

- CF - Returns all records for the cluster caching facilities in the current Db2 pureScale instance
- MEMBER - Returns all records for the members in the current database instance.

If this parameter is null or an empty string, all records returned.

*db\_partition\_num*

An optional input argument of type INTEGER that specifies a valid database partition number. Information is returned for all database partition numbers that match the input. If this parameter is null or -2, all records are returned. If this parameter is -1, information for the currently connected member is returned. For a Db2 pureScale instance, the only valid inputs are 0 or null; all other input values will not return data.

## Authorization

EXECUTE privilege on the DB2\_GET\_INSTANCE\_INFO table function.

## Default PUBLIC privilege

None

## Examples

The following example uses the DB2\_GET\_INSTANCE\_INFO table function to retrieve information from a Db2 pureScale instance with four members and two cluster caching facilities. In this example, member number 2 is being restarted on the host called so1:

```
SELECT * FROM TABLE(DB2_GET_INSTANCE_INFO(null,','',',null)) as T
```

The following is an example of output from this query.

ID	HOME_HOST	CURRENT_HOST	TYPE	STATE	ALERT	DB_P..._NUM	LOGICAL_PORT	NETNAME
0	so1	so1	MEMBER	STARTED	NO	0	0	so1-ib0
1	so2	so1	MEMBER	RESTARTING	NO	0	0	so2-ib0
2	so3	so3	MEMBER	STARTED	NO	0	0	so3-ib0
3	so4	so4	MEMBER	STARTED	NO	0	0	so4-ib0
128	so5	so5	CF	PRIMARY	NO	-	-	so5-ib0
129	so6	so6	CF	PEER	NO	-	-	so6-ib0

6 record(s) selected.

The following example uses the DB2\_GET\_INSTANCE\_INFO table function to retrieve information from a partitioned database instance with four members using the High Availability Disaster Recovery (HADR) feature:

```
SELECT * FROM TABLE(DB2_GET_INSTANCE_INFO(null,'','','null')) as T
```

The following is an example of output from this query.

ID	HOME_HOST	CURRENT_HOST	TYPE	STATE	ALERT	DB_PARTITION_NUM	LOGICAL_PORT	NETNAME
0	so1	so1	MEMBER	STARTED	NO	0	0	so1-ib0
2	so2	so2	MEMBER	STARTED	NO	2	0	so2-ib0
4	so3	so3	MEMBER	STARTED	NO	4	0	so3-ib0
7	so4	so4	MEMBER	STARTED	NO	7	0	so4-ib0

4 record(s) selected.

## Information returned

Table 64. Information returned for DB2\_MEMBER administrative view

Column name	Data type	Description
ID	SMALLINT	id - cluster caching facility identification monitor element
HOME_HOST	VARCHAR(255)	The machine which was associated with the member when it was first added to the instance.
CURRENT_HOST	VARCHAR(255)	The machine name on which the member is currently running.
STATE	VARCHAR(32)	<p>The state of the member or cluster caching facility.</p> <p>The potential states for a member are STARTED, STOPPED, RESTARTING, WAITING_FOR_FAILBACK, ERROR, and UNKNOWN.</p> <p>The potential states for a cluster caching facility include STOPPED, RESTARTING, BECOMING_PRIMARY, PRIMARY, CATCHUP<sup>1</sup>, PEER, ERROR, and UNKNOWN.</p> <p>The NULL state for a member or a cluster caching facility indicates that state is inapplicable. For example, if in a partitioned database environment.</p> <p>See Values for member and cluster caching facility states and alerts for more information.</p>

Table 64. Information returned for DB2\_MEMBER administrative view (continued)

Column name	Data type	Description
ALERT	VARCHAR(8)	<p>Information about alerts on the instance:</p> <ul style="list-style-type: none"> <li>• YES - there is an alert</li> <li>• NO - there are no alerts</li> <li>• The NULL state for a member or a cluster caching facility indicates that state is inapplicable. For example, if in a partitioned database environment.</li> </ul> <p>Use the DB2_INSTANCE_ALERTS administrative view to obtain more information about an alert.</p>

Table 65. Information returned for DB2\_CF administrative view

Column name	Data type	Description
ID	SMALLINT	id - cluster caching facility identification monitor element
CURRENT_HOST	VARCHAR(255)	The machine name on which the member is currently running.
STATE	VARCHAR(32)	<p>The state of the member or cluster caching facility.</p> <p>The potential states for a member are STARTED, STOPPED, RESTARTING, WAITING_FOR_FAILBACK, ERROR, and UNKNOWN.</p> <p>The potential states for a cluster caching facility include STOPPED, RESTARTING, BECOMING_PRIMARY, PRIMARY, CATCHUP<sup>1</sup>, PEER, ERROR, and UNKNOWN.</p> <p>The NULL state for a member or a cluster caching facility indicates that state is inapplicable. For example, if in a partitioned database environment.</p> <p>See Values for member and cluster caching facility states and alerts for more information.</p>
ALERT	VARCHAR(8)	<p>Information about alerts on the instance:</p> <ul style="list-style-type: none"> <li>• YES - there is an alert</li> <li>• NO - there are no alerts</li> <li>• The NULL state for a member or a cluster caching facility indicates that state is inapplicable. For example, if in a partitioned database environment.</li> </ul> <p>Use the DB2_INSTANCE_ALERTS administrative view to obtain more information about an alert.</p>

Table 66. Information returned for DB2\_GET\_INSTANCE\_INFO table function

Column name	Data type	Description
ID	SMALLINT	id - cluster caching facility identification monitor element
HOME_HOST	VARCHAR(255)	The machine which was associated with the member when it was first added to the instance.
CURRENT_HOST	VARCHAR(255)	The machine name on which the member is currently running.
TYPE	VARCHAR(32)	Either 'MEMBER' or 'CF'.

Table 66. Information returned for DB2\_GET\_INSTANCE\_INFO table function (continued)

Column name	Data type	Description
STATE	VARCHAR(32)	<p>The state of the member or cluster caching facility.</p> <p>The potential states for a member are STARTED, STOPPED, RESTARTING, WAITING_FOR_FAILBACK, ERROR, and UNKNOWN.</p> <p>The potential states for a cluster caching facility include STOPPED, RESTARTING, BECOMING_PRIMARY, PRIMARY, CATCHUP<sup>1</sup>, PEER, ERROR, and UNKNOWN.</p> <p>The NULL state for a member or a cluster caching facility indicates that state is inapplicable. For example, if in a partitioned database environment.</p> <p>See Values for member and cluster caching facility states and alerts for more information.</p>
ALERT	VARCHAR(8)	<p>Information about alerts on the instance:</p> <ul style="list-style-type: none"> <li>• YES - there is an alert</li> <li>• NO - there are no alerts</li> <li>• The NULL state for a member or a cluster caching facility indicates that state is inapplicable. For example, if in a partitioned database environment.</li> </ul> <p>Use the DB2_INSTANCE_ALERTS administrative view to obtain more information about an alert.</p>
DB_PARTITION_NUM	SMALLINT	The database partition number for this member.
LOGICAL_PORT	SMALLINT	The logical port number of the member or cluster caching facility.
NETNAME	VARCHAR(255)	The name of the high speed interconnect for internal database communications.

**Note:**

1. CATCHUP includes a percentage value as part of the returned state. This percentage value represents the amount to which the secondary cluster caching facility has caught up to the current state of the primary caching facility.

## Environment routines and views

This grouping of routines and views can be used to retrieve information about a database environment.

### ENV\_CF\_SYS\_RESOURCES administrative view - Get cluster caching facility system resource information

The ENV\_CF\_SYS\_RESOURCES administrative view returns a list of system resources used by the cluster caching facilities (also known as CFs) on the system.

The schema is SYSIBMADM.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the view
- CONTROL privilege on the view

- DATAACCESS authority

### Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

### Examples

The following example demonstrates the information you can obtain using the ENV\_CF\_SYS\_RESOURCES administrative view.

```
SELECT * FROM SYSIBMADM.ENV_CF_SYS_RESOURCES
```

### Information returned

Table 67. Information returned by the ENV\_CF\_SYS\_RESOURCES administrative view

Column name	Data type	Description
NAME	VARCHAR(128)	Name of the system resource, see following table for possible values.
VALUE	VARCHAR(1024)	The value of the system resource.
DATATYPE	VARCHAR(128)	The data type of the value, see following table for possible values.
UNIT	VARCHAR(128)	The unit represented by the value, if applicable; null otherwise.
ID	SMALLINT	id - cluster caching facility identification monitor element

Table 68. Possible values for the NAME and DATATYPE columns

Information type	Name	Data Types	Description
Operating system	HOST_NAME	VARCHAR(256)	Name of the host the cluster caching facility is running on.
Physical memory	MEMORY_TOTAL	BIGINT	Total size of physical memory.
	MEMORY_FREE	BIGINT	Amount of free physical memory.
	MEMORY_SWAP_TOTAL	BIGINT	Total amount of swap space.
	MEMORY_SWAP_FREE	BIGINT	Amount of free swap space.
Virtual memory	VIRTUAL_MEM_TOTAL	BIGINT	Total amount of virtual memory on the system.
	VIRTUAL_MEM_RESERVED	BIGINT	Amount of reserved virtual memory.
CPU load	CPU_USAGE_TOTAL	DECIMAL	Percentage of time the CF workers are busy doing work when the CF is bound to its own set of CPUs

## ENV\_FEATURE\_INFO administrative view - Return license information for product offerings

The ENV\_FEATURE\_INFO administrative view returns information about all available offerings for which a license is required. For each offering, there is information about whether or not a valid license for the offering has been installed.

The schema is SYSIBMADM.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the ENV\_FEATURE\_INFO administrative view
- CONTROL privilege on the ENV\_FEATURE\_INFO administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

### Example

Request the installed product offerings license information.

```
SELECT FEATURE_NAME, FEATURE_FULLNAME, LICENSE_INSTALLED,
 PRODUCT_NAME, FEATURE_USE_STATUS
FROM SYSIBMADM.ENV_FEATURE_INFO
```

The following is an example of output from this query.

```
FEATURE_NAME FEATURE_FULLNAME ...

BAF IBM_DB2_BLU_ACCELERATION_IN_MEMORY_OFFERING ...
PMF IBM_DB2_PERFORMANCE_MANAGEMENT_OFFERING ...
EF IBM_DB2_ENCRYPTION_OFFERING ...
BCF IBM_DB2_BUSINESS_APPLICATION_CONTINUITY_OFFERING...
```

Output from this query (continued).

```
... LICENSE_INSTALLED PRODUCT_NAME FEATURE_USE_STATUS
... -----
... Y ESE NOT_USED
... N ESE NOT_USED
... N ESE NOT_USED
... N ESE NOT_USED
```

### ENV\_FEATURE\_INFO administrative view metadata

Table 69. ENV\_FEATURE\_INFO administrative view metadata

Column name	Data type	Description
FEATURE_NAME	VARCHAR(26)	Short names for product offerings that are available on licensed servers.
FEATURE_FULLNAME	VARCHAR(100)	Full name of product offerings. Column values are displayed in English using uppercase letters. Words are separated by an underscore character instead of a space character.

Table 69. ENV\_FEATURE\_INFO administrative view metadata (continued)

Column name	Data type	Description
LICENSE_INSTALLED	CHAR(1)	Indicates if offering is licensed. If the value is 'N', the offering is not licensed. If the value is 'Y', the offering is licensed.
PRODUCT_NAME	VARCHAR(26)	product_name - Product Name monitor element
FEATURE_USE_STATUS	VARCHAR(30)	Indicates the license compliance status. This value indicates the usage status of the offering. There are three possible values: <ul style="list-style-type: none"> <li>• IN_COMPLIANCE: Offering has been used at least once and there is a valid license for the feature.</li> <li>• IN_VIOLATION: Offering has been used at least once and there is no valid license for the feature.</li> <li>• NOT_USED: Offering has not been used.</li> </ul>

## ENV\_GET\_DB2\_EDU\_SYSTEM\_RESOURCES table function - Return information about the database engine dispatchable units

The ENV\_GET\_DB2\_EDU\_SYSTEM\_RESOURCES table function returns system resource metrics at the instance level for all database engine dispatchable units (EDUs).

### Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- SQLADM authority
- DBADM authority

### Default PUBLIC privilege

None

### Syntax

►►—ENV\_GET\_DB2\_EDU\_SYSTEM\_RESOURCES—(—*member*—)—————►►

The schema is SYSPROC.

### Routine parameters

*member*

An input parameter of type INTEGER that specifies a valid database member from which the records are to be fetched. Specify -1 or NULL for the current member, or -2 for information from all active database members. An active database member is a database that is available for connection and use by applications. The default value is NULL.

## Information returned

Table 70. Information returned by the ENV\_GET\_DB2\_EDU\_SYSTEM\_RESOURCES routine

Column name	Data type	Description
MEMBER	SMALLINT	member - Database member monitor element
EDU_ID	BIGINT	edu_ID - Engine dispatchable unit ID monitor element
EDU_NAME	VARCHAR (128)	edu_name - Engine dispatchable unit name monitor element
DB2_PROCESS_ID	BIGINT	db2_process_id - database process ID monitor element
TID	BIGINT	agent_tid - Agent thread ID monitor element
KTID	BIGINT	ktid - Kernel thread ID monitor element
CPU_USER_TIME_MS	BIGINT	CPU user processing time in microseconds.
CPU_SYSTEM_TIME_MS	BIGINT	CPU system processing time in microseconds.
RESOURCE_METHOD	VARCHAR (8)	Resource policy method in use for this EDU.
RESOURCE_BINDING	VARCHAR (30)	Resource policy binding that is in use for this EDU.
RESOURCE_GROUP	VARCHAR (30)	Resource group that is associated with the resource policy binding that is in use for this EDU.

## Examples

- List the EDU\_NAME and DB2\_PROCESS\_ID on all members.

```
SELECT EDU_NAME, DB2_PROCESS_ID
FROM TABLE (ENV_GET_DB2_EDU_SYSTEM_RESOURCES(-2))
```

This query returns the following output:

EDU_NAME	DB2_PROCESS_ID
db2agntdp (TEST)	4615
db2agntdp (TEST)	4615
db2agent (TEST)	4615
db2evmli (DB2DETAILDEADLOCK)	4615
db2pcsd (TEST)	4615
db2fw3 (TEST)	4615
db2fw2 (TEST)	4615
db2fw1 (TEST)	4615
db2fw0 (TEST)	4615
db2lused (TEST)	4615
db2w1md (TEST)	4615
db2taskd (TEST)	4615
db2pfchr (TEST)	4615
db2pfchr (TEST)	4615
db2pfchr (TEST)	4615
db2pclnr (TEST)	4615

db2pclnr (TEST)	4615
db2pclnr (TEST)	4615
db2pclnr (TEST)	4615
db2lfr.0 (TEST)	4615
db2loggw (TEST)	4615
db2loggr (TEST)	4615
db2dlock (TEST)	4615
db2agent (TEST)	4615
db2stmm (TEST)	4615
db2resync	4615
db2ipccm	4615
db2w1mtm	4615
db2w1mt	4615
db2licc	4615
db2thcln	4615
db2alarm	4615
db2sysc	4615

33 record(s) selected.

- List TID, RESOURCE\_METHOD, and CPU\_USER\_TIME\_MS on all members.

```
SELECT TID, RESOURCE_METHOD, CPU_USER_TIME_MS
FROM TABLE (ENV_GET_DB2_EDU_SYSTEM_RESOURCES(-2))
```

This query returns the following output:

TID	RESOURCE_METHOD	CPU_USER_TIME_MS
-----	-----	-----
46913046964544	-	6970000
46913063741760	-	14950000
46913025993024	-	1260000
46913042770240	-	0
46913030187328	-	0
46913114073408	-	0
46913059547456	-	0
46913038575936	-	0
46913055353152	-	0
46913067936064	-	60000
46913072130368	-	0
46913051158848	-	0
46913080518976	-	0
46913076324672	-	20000
46913084713280	-	20000
46913097296192	-	0
46913101490496	-	0
46913088907584	-	0
46913109879104	-	0
46913105684800	-	0
46913093101888	-	60000
46913118267712	-	70000
46913122462016	-	0
46913034381632	-	4070000
46913126656320	-	7000000
46913130850624	-	320000
46913135044928	-	50000
46913139239232	-	0
46913143433536	-	0
46913147627840	-	0
46913151822144	-	0
46913156016448	-	0
46912895969600	-	7660000

33 record(s) selected.

## ENV\_GET\_DB2\_SYSTEM\_RESOURCES table function - Return database system information

The ENV\_GET\_DB2\_SYSTEM\_RESOURCES table function returns CPU usage and database process information for specified members in the current instance.

### Syntax

►►—ENV\_GET\_DB2\_SYSTEM\_RESOURCES—(—*member*—)—————►►

The schema is SYSPROC.

### Table function parameters

#### *member*

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all database members. If the NULL value is specified, -1 is set implicitly.

### Authorization

One of the following authorizations is required:

- EXECUTE privilege on the table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Example

```
SELECT VARCHAR(db2_process_name, 20) AS NAME, CPU_USER, CPU_SYSTEM
 FROM TABLE(ENV_GET_DB2_SYSTEM_RESOURCES(-2))
```

This query returns the following output:

NAME	CPU_USER	CPU_SYSTEM
db2fmp	14	9
db2sysc	11752	541
db2syscr	13	8
db2fmp	24	14

4 record(s) selected.

### Information returned

Table 71. Information returned by the ENV\_GET\_DB2\_SYSTEM\_RESOURCES table function

Column name	Data type	Description
MEMBER	SMALLINT	member - Database member monitor element
DB2_PROCESS_NAME	VARCHAR(128)	db2_process_name - process name monitor element

Table 71. Information returned by the ENV\_GET\_DB2\_SYSTEM\_RESOURCES table function (continued)

Column name	Data type	Description
DB2_PROCESS_ID	BIGINT	db2_process_id - process ID monitor element
CPU_USER	BIGINT	cpu_user - Non-kernel processing time monitor element
CPU_SYSTEM	BIGINT	cpu_system - Kernel time monitor element

## ENV\_GET\_NETWORK\_RESOURCES table function - Return network information

The ENV\_GET\_NETWORK\_RESOURCES table function returns information for all active network adaptors on the host machines on which the database system runs.

### Authorization

One of the following authorizations is required:

- EXECUTE privilege on the table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

The schema is SYSPROC.

### Default PUBLIC privilege

None

### Example

```
SELECT varchar(adapter_name, 20) as name,
 packets_received,
 packets_sent,
 total_bytes_received,
 total_bytes_sent
FROM TABLE(ENV_GET_NETWORK_RESOURCES())
```

The query returns the following input:

NAME	PACKETS_RECEIVED	PACKETS_SENT	TOTAL_BYTES_RECEIVED
lo	467182039	467182039	528451011980
eth0	426287355	431398744	351656704796
eth1	0	0	0
TOTAL_BYTES_SENT			
	528451011980		
	272061746005		
	0		

## Information returned

Table 72. Information returned by the ENV\_GET\_NETWORK\_RESOURCES table function

Column name	Data type	Description
MEMBER	SMALLINT	member - Database member monitor element
HOST_NAME	VARCHAR(255)	host_name - Host name monitor element
ADAPTER_NAME	VARCHAR(255)	adapter_name - Adapter name monitor element
PACKETS_RECEIVED	BIGINT	packets_received - Packets received monitor element
PACKETS_SENT	BIGINT	packets_sent - Packets sent monitor element
PACKET_RECEIVE_ERRORS	BIGINT	packet_receive_errors - Packet receive errors monitor element
PACKET_SEND_ERRORS	BIGINT	packet_send_errors - Packet send errors monitor element
TOTAL_BYTES_RECEIVED	BIGINT	total_bytes_received - Total bytes received monitor element
TOTAL_BYTES_SENT	BIGINT	total_bytes_sent - Total bytes sent monitor element

## ENV\_GET\_REG\_VARIABLES table function - Retrieve database registry settings that are in use

The ENV\_GET\_REG\_VARIABLES table function returns the database registry settings from one or all database members.

**Note:** The ENV\_GET\_REG\_VARIABLES table function replaces the deprecated REG\_VARIABLES administrative view. The ENV\_GET\_REG\_VARIABLES function is different in that it allows for a single parameter value to denote a specific member to query, and returns an addition result for the registry setting currently stored on disk.

## Authorization

One of the following authorizations is required:

- EXECUTE privilege on the ENV\_GET\_REG\_VARIABLES table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, one of the following privileges or authorities is also required:

- EXECUTE privilege on the ENV\_GET\_REG\_VARIABLES table function

## Default PUBLIC privilege

None

## Syntax

►►—ENV\_GET\_REG\_VARIABLES—(—*member*—,—*showall*—)—►►

The schema is SYSPROC.

### Table function parameters

#### *member*

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all database members. If the NULL value is specified, -1 is set implicitly.

#### *showall*

An input argument of type SMALLINT that specifies if information on all registry variables is returned. Specify 0 to show only those registry variables currently set, or 1 to show all registry variables. If the null value is specified, 0 is set implicitly. The default value is NULL.

### Information returned

Table 73. Information returned by the ENV\_GET\_REG\_VARIABLES table function

Column name	Data type	Description
MEMBER	SMALLINT	member - Database member monitor element
REG_VAR_NAME	VARCHAR(256)	Name of the database registry variable.
REG_VAR_VALUE	VARCHAR(2048)	Current setting of the database registry variable in memory.
REG_VAR_ON_DISK_VALUE	VARCHAR(2048)	Current setting of the database registry variable on disk.
IS_AGGREGATE	SMALLINT	Indicates whether or not the database registry variable is an aggregate variable. The possible return values are 0 if it is not an aggregate variable, and 1 if it is an aggregate variable.
AGGREGATE_NAME	VARCHAR(256)	Name of the aggregate if the database registry variable is currently obtaining its value from a configured aggregate. If the registry variable is not being set through an aggregate, or is set through an aggregate but has been overridden, the value of AGGREGATE_NAME is NULL.

Table 73. Information returned by the ENV\_GET\_REG\_VARIABLES table function (continued)

Column name	Data type	Description
LEVEL	CHAR(1)	Indicates the level at which the database registry variable acquires its value. The possible return values and the corresponding levels that they represent are: <ul style="list-style-type: none"> <li>• I = instance</li> <li>• G = global</li> <li>• N = database partition</li> <li>• E = environment</li> </ul>
REG_VAR_DEFAULT_VALUE	VARCHAR(2048)	Default value for the registry variable.

### Examples

- Show all database registry settings (including registry variables that are not set) for all database members:

```
select MEMBER, REG_VAR_NAME,
 REG_VAR_VALUE, REG_VAR_ON_DISK_VALUE
from table(ENV_GET_REG_VARIABLES(-2, 1))
```

- In this example, the registry variable DB2DBDFT, which specifies the database alias name to use for implicit connections, is set to CORP\_1. This is done before the database instance starts:

```
db2set db2dbdft=CORP_1
db2start
```

Then you can issue a query to show that registry variable setting:

```
select substr(reg_var_value,1,20) as VALUE,
 substr(reg_var_on_disk_value,1,20) as ON_DISK_VALUE
from table(env_get_reg_variables(-1))
where reg_var_name = 'DB2DBDFT'
```

This query returns the following output:

```
VALUE ON_DISK_VALUE

CORP_1 CORP_1
```

1 record(s) selected.

To demonstrate the difference between in memory and on disk values for the registry settings, the DB2DBDFT registry variable is then altered:

```
db2set db2dbdft=DEPT_MAJOR
```

Running the same query as the previous one, shows that this new value is immediately picked up by the ENV\_GET\_REG\_VARIABLES function:

```
VALUE ON_DISK_VALUE

CORP_1 DEPT_MAJOR
```

1 record(s) selected.

However, note that the in memory value will not change to the new value until the database instance is restarted.

## ENV\_GET\_INSTANCE\_CODE\_LEVELS table function - Return architectural level

The ENV\_GET\_INSTANCE\_CODE\_LEVELS table function returns the code level for the instance, members, and cluster caching facilities.

### Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the routine is automatically created.

### Syntax

►►—ENV\_GET\_INSTANCE\_CODE\_LEVELS—(—)—►►

The schema is SYSPROC.

### Information returned

Table 74. Information returned by the ENV\_GET\_INSTANCE\_CODE\_LEVELS routine

Column name	Data type	Description
RECORD_TYPE	VARCHAR(10)	The record type, which specifies if rolling updates information is specific to an instance, cluster caching facility, or member.
ID	SMALLINT	Host as represented in db2nodes.cfg (0 for the INSTANCE level record).
ARCHITECTURE_LEVEL_DISK	VARCHAR(50)	Architecture level value stored on disk.
ARCHITECTURE_LEVEL_MEMORY	VARCHAR(50)	Architecture level value stored in memory.
SECTION_LEVEL_DISK	VARCHAR(50)	Section level stored on disk.
SECTION_LEVEL_MEMORY	VARCHAR(50)	Section level stored in memory.
CODE_LEVEL_DISK	VARCHAR(50)	Code level stored on disk.
CODE_LEVEL_MEMORY	VARCHAR(50)	Code level stored in memory.

Table 74. Information returned by the ENV\_GET\_INSTANCE\_CODE\_LEVELS routine (continued)

Column name	Data type	Description
STATE	VARCHAR(60)	Rolling Updates state: <ul style="list-style-type: none"> <li>NONE</li> <li>NOT COMPUTED</li> <li>OFFLINE UPDATE INITIATED</li> <li>ONLINE UPDATE INITIATED</li> </ul>
LAST_UPDATED	TIMESTAMP	Timestamp indicating when this member was last updated.
ARCHITECTURE_LEVEL_DISK_PACKED	VARCHAR(8) FOR BIT DATA	Packed format of the architecture level stored on disk.
ARCHITECTURE_LEVEL_MEMORY_PACKED	VARCHAR(8) FOR BIT DATA	Packed format of the architecture level stored in memory.
SECTION_LEVEL_DISK_PACKED	VARCHAR(8) FOR BIT DATA	Packed format of the section level stored on disk.
SECTION_LEVEL_MEMORY_PACKED	VARCHAR(8) FOR BIT DATA	Packed format of the section level stored in memory.
CODE_LEVEL_DISK_PACKED	VARCHAR(8) FOR BIT DATA	Packed format of the code level stored on disk.
CODE_LEVEL_MEMORY_PACKED	VARCHAR(8) FOR BIT DATA	Packed format of the code level stored in memory.

## Example

Simulation of an online rolling update from Version 10.5 Fix Pack 1 to Fix Pack 2 for an instance named "INST1". It is assumed that Fix Pack 2 introduced a new architecture level but the section level remained unchanged since Version 10.5.

1. The instance starts in a homogeneous state. All members are identical.

```
SELECT RECORD_TYPE, ID, SUBSTR(ARCHITECTURE_LEVEL_DISK,1,28)
 AS ARCHITECTURE_LEVEL, SUBSTR(SECTION_LEVEL_DISK,1,26)
 AS SECTION_LEVEL, SUBSTR(CODE_LEVEL_DISK,1, 28)
 AS CODE_LEVEL, LAST_UPDATED
FROM TABLE (SYSPROC.ENV_GET_INSTANCE_CODE_LEVELS())
```

This query returns the following output:

RECORD_TYPE	ID	ARCHITECTURE_LEVEL	SECTION_LEVEL	...
INSTANCE	-	V:10 R:5 M:0 F:1 I:0 SB:0	-	...
MEMBER	0	V:10 R:5 M:0 F:1 I:0 SB:0	V:10 R:5 M:0 F:0 I:0 SB:0	...
MEMBER	1	V:10 R:5 M:0 F:1 I:0 SB:0	V:10 R:5 M:0 F:0 I:0 SB:0	...
MEMBER	2	V:10 R:5 M:0 F:1 I:0 SB:0	V:10 R:5 M:0 F:0 I:0 SB:0	...
CF	128	V:10 R:5 M:0 F:1 I:0 SB:0	-	...
CF	129	V:10 R:5 M:0 F:1 I:0 SB:0	-	...

Query output (continued):

...CODE_LEVEL	LAST_UPDATED
...V:10 R:5 M:0 F:1 I:0 SB:0	2012-10-06-19.12.00.000000

```

...V:10 R:5 M:0 F:1 I:0 SB:0 2012-10-06-19.12.00.000000
...V:10 R:5 M:0 F:1 I:0 SB:0 2012-10-06-19.12.00.000000
...V:10 R:5 M:0 F:1 I:0 SB:0 2012-10-06-19.12.00.000000
...V:10 R:5 M:0 F:1 I:0 SB:0 2012-10-06-19.12.00.000000
...V:10 R:5 M:0 F:1 I:0 SB:0 2012-10-06-19.12.00.000000

```

6 record(s) selected.

- The user updates two cluster caching facilities (CFs) using the following command on both CFs:

```
./installFixPack -p FP2_install_path INST1
```

Query output:

RECORD_TYPE	ID	ARCHITECTURE_LEVEL	SECTION_LEVEL	...
INSTANCE	-	V:10 R:5 M:0 F:1 I:0 SB:0	-	...
MEMBER	0	V:10 R:5 M:0 F:1 I:0 SB:0	V:10 R:5 M:0 F:0 I:0 SB:0	...
MEMBER	1	V:10 R:5 M:0 F:1 I:0 SB:0	V:10 R:5 M:0 F:0 I:0 SB:0	...
MEMBER	2	V:10 R:5 M:0 F:1 I:0 SB:0	V:10 R:5 M:0 F:0 I:0 SB:0	...
CF	128	<b>V:10 R:5 M:0 F:2 I:0 SB:0</b>	-	...
CF	129	<b>V:10 R:5 M:0 F:2 I:0 SB:0</b>	-	...

Query output (continued):

...	CODE_LEVEL	LAST_UPDATED
...	V:10 R:5 M:0 F:1 I:0 SB:0	2012-10-06-19.12.00.000000
...	V:10 R:5 M:0 F:1 I:0 SB:0	2012-10-06-19.12.00.000000
...	V:10 R:5 M:0 F:1 I:0 SB:0	2012-10-06-19.12.00.000000
...	V:10 R:5 M:0 F:1 I:0 SB:0	2012-10-06-19.12.00.000000
...	<b>V:10 R:5 M:0 F:2 I:0 SB:0</b>	<b>2012-10-06-19.23.44.000000</b>
...	<b>V:10 R:5 M:0 F:2 I:0 SB:0</b>	<b>2012-10-06-19.22.49.000000</b>

6 record(s) selected.

- The three members are subsequently moved to Fix Pack 2 using the same installFixPack command:

```
./installFixPack -p FP2_install_path INST1
```

Query output:

RECORD_TYPE	ID	ARCHITECTURE_LEVEL	SECTION_LEVEL	...
INSTANCE	-	V:10 R:5 M:0 F:1 I:0 SB:0	-	...
MEMBER	0	<b>V:10 R:5 M:0 F:2 I:0 SB:0</b>	<b>V:10 R:5 M:0 F:0 I:0 SB:0</b>	...
MEMBER	1	<b>V:10 R:5 M:0 F:2 I:0 SB:0</b>	<b>V:10 R:5 M:0 F:0 I:0 SB:0</b>	...
MEMBER	2	<b>V:10 R:5 M:0 F:2 I:0 SB:0</b>	<b>V:10 R:5 M:0 F:0 I:0 SB:0</b>	...
CF	128	V:10 R:5 M:0 F:2 I:0 SB:0	-	...
CF	129	V:10 R:5 M:0 F:2 I:0 SB:0	-	...

Query output (continued):

...	CODE_LEVEL	LAST_UPDATED
...	V:10 R:5 M:0 F:1 I:0 SB:0	2012-10-06-19.12.00.000000
...	<b>V:10 R:5 M:0 F:2 I:0 SB:0</b>	<b>2012-10-06-19.29.39.000000</b>
...	<b>V:10 R:5 M:0 F:2 I:0 SB:0</b>	<b>2012-10-06-19.30.34.000000</b>
...	<b>V:10 R:5 M:0 F:2 I:0 SB:0</b>	<b>2012-10-06-19.31.25.000000</b>
...	V:10 R:5 M:0 F:2 I:0 SB:0	2012-10-06-19.23.44.000000
...	V:10 R:5 M:0 F:2 I:0 SB:0	2012-10-06-19.22.49.000000

6 record(s) selected.

- The cluster is then committed at the new level of code, using the following command:

```
./installFixPack -commit_new_level
```

Query output:

RECORD_TYPE	ID	ARCHITECTURE_LEVEL	SECTION_LEVEL	...
<b>INSTANCE</b>	-	<b>V:10 R:5 M:0 F:2 I:0 SB:0</b>	-	...
MEMBER	0	V:10 R:5 M:0 F:2 I:0 SB:0	V:10 R:5 M:0 F:0 I:0 SB:0	...
MEMBER	1	V:10 R:5 M:0 F:2 I:0 SB:0	V:10 R:5 M:0 F:0 I:0 SB:0	...
MEMBER	2	V:10 R:5 M:0 F:2 I:0 SB:0	V:10 R:5 M:0 F:0 I:0 SB:0	...
CF	128	V:10 R:5 M:0 F:2 I:0 SB:0	-	...
CF	129	V:10 R:5 M:0 F:2 I:0 SB:0	-	...

Query output (continued):

...	CODE_LEVEL	LAST_UPDATED
...	...	...
...	<b>V:10 R:5 M:0 F:2 I:0 SB:0</b>	<b>2012-10-06-19.36.40.000000</b>
...	V:10 R:5 M:0 F:2 I:0 SB:0	2012-10-06-19.29.39.000000
...	V:10 R:5 M:0 F:2 I:0 SB:0	2012-10-06-19.30.34.000000
...	V:10 R:5 M:0 F:2 I:0 SB:0	2012-10-06-19.31.25.000000
...	V:10 R:5 M:0 F:2 I:0 SB:0	2012-10-06-19.23.44.000000
...	V:10 R:5 M:0 F:2 I:0 SB:0	2012-10-06-19.22.49.000000

6 record(s) selected.

## ENV\_GET\_SYSTEM\_RESOURCES table function - Return system information

The ENV\_GET\_SYSTEM\_RESOURCES table function returns operating system, CPU, memory, and other information that is related to members on the system.

The active database can reside on one or more members on the system. This table function returns data only from members where the database that issued the command is active.

**Note:** The ENV\_GET\_SYSTEM\_RESOURCES table function replaces the ENV\_GET\_SYS\_RESOURCES table function and the associated ENV\_SYS\_RESOURCES administrative view.

The ENV\_GET\_SYSTEM\_RESOURCES table function returns all data in one row with multiple columns for each member, instead of multiple rows for each member. Additionally, data that the ENV\_GET\_SYS\_RESOURCES table function returns for the DBPARTITIONNUM column is returned in the MEMBER column by the ENV\_GET\_SYSTEM\_RESOURCES table function.

The schema is SYSPROC.

## Authorization

One of the following authorizations is required:

- EXECUTE privilege on the table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

None

## Information returned

Table 75. Information returned for ENV\_GET\_SYSTEM\_RESOURCES

Column name	Data type	Description
MEMBER	SMALLINT	member - Database member monitor element
OS_NAME	VARCHAR(256)	os_name - Operating system name monitor element
HOST_NAME	VARCHAR(255)	host_name - Host name monitor element
OS_VERSION	VARCHAR(256)	os_version - Operating system version monitor element
OS_RELEASE	VARCHAR(256)	os_release - Operating system release monitor element
MACHINE_IDENTIFICATION	VARCHAR(256)	machine_identification - Host hardware identification monitor element
OS_LEVEL	VARCHAR(256)	os_level - Operating system level monitor element
CPU_TOTAL	BIGINT	cpu_total - Number of CPUs monitor element
CPU_ONLINE	BIGINT	cpu_online - Number of CPUs online monitor element
CPU_CONFIGURED	BIGINT	cpu_configured - Number of configured CPUs monitor element
CPU_SPEED	BIGINT	cpu_speed - CPU clock speed monitor element
CPU_TIMEBASE	BIGINT	cpu_timebase - Frequency of timebase register increment monitor element
CPU_HMT_DEGREE	BIGINT	cpu_hmt_degree - Number of logical CPUs monitor element
CPU_CORES_PER_SOCKET	BIGINT	cpu_cores_per_socket - Number of CPU cores per socket monitor element
MEMORY_TOTAL	BIGINT	memory_total - Total physical memory monitor element
MEMORY_FREE	BIGINT	memory_free - Amount of free physical memory monitor element
MEMORY_SWAP_TOTAL	BIGINT	memory_swap_total - Total swap space monitor element
MEMORY_SWAP_FREE	BIGINT	memory_swap_free - Total free swap space monitor element
VIRTUAL_MEM_TOTAL	BIGINT	virtual_mem_total - Total virtual memory monitor element
VIRTUAL_MEM_RESERVED	BIGINT	virtual_mem_reserved - Reserved virtual memory monitor element
VIRTUAL_MEM_FREE	BIGINT	virtual_mem_free - Free virtual memory monitor element
CPU_LOAD_SHORT	DOUBLE	cpu_load_short - Processor load (short timeframe) monitor element
CPU_LOAD_MEDIUM	DOUBLE	cpu_load_medium - Processor load (medium timeframe) monitor element
CPU_LOAD_LONG	DOUBLE	cpu_load_long - Processor load (long timeframe) monitor element
CPU_USAGE_TOTAL	SMALLINT	cpu_usage_total - Processor usage monitor element
CPU_USER <sup>1</sup>	BIGINT	cpu_user - Non-kernel processing time monitor element
CPU_IDLE <sup>1</sup>	BIGINT	cpu_idle - Processor idle time monitor element
CPU_IOWAIT <sup>1</sup>	BIGINT	cpu_iowait - IO Wait time monitor element
CPU_SYSTEM <sup>1</sup>	BIGINT	cpu_system - Kernel time monitor element
SWAP_PAGE_SIZE	BIGINT	swap_page_size - Swap page size monitor element

Table 75. Information returned for ENV\_GET\_SYSTEM\_RESOURCES (continued)

Column name	Data type	Description
SWAP_PAGES_IN	BIGINT	swap_pages_in - Pages swapped in from disk monitor element
SWAP_PAGES_OUT	BIGINT	swap_pages_out - Pages swapped out to disk monitor element
OS_FULL_VERSION	VARCHAR(256)	os_full_version - Operating system full version monitor element
OS_KERNEL_VERSION	VARCHAR(256)	os_kernel_version - Operating system kernel identifier monitor element
OS_ARCH_TYPE	VARCHAR(256)	os_arch_type - Operating system architecture type monitor element
<b>Note:</b> <b>1</b> These metrics have been aggregated across all logical processors on the system. On the AIX operating system, the metrics are for the Workload partition (WPAR) and Logical partition (LPAR) on which the database server is running.		

### Usage notes

- In a virtualized environment (such as VMWare ESX) the database system might be unable to determine the CPU topology such as CPU\_HMT\_DEGREE or CPU\_CORES\_PER\_SOCKET. This occurs because of limited information that the hypervisor makes available to the virtualized operating system. In such cases, CPU\_CORES\_PER\_SOCKET and CPU\_HMT\_DEGREE will have a value of 1.
- To return accurate results, the ENV\_GET\_SYSTEM\_RESOURCES table function takes more than 1 second on AIX, HP, and Windows operating systems.

### Examples

*Example 1:* Obtain operating system information for every member in a three-member Db2 pureScale setup:

```
select MEMBER, varchar(HOST_NAME,12) as HOST_NAME, varchar(OS_NAME,8) as OS_NAME,
 varchar(OS_VERSION,8) as OS_VERSION, varchar(OS_RELEASE,8)
 as OS_RELEASE from table(SYSPROC.ENV_GET_SYSTEM_RESOURCES()) order by MEMBER
```

Sample output is as follows:

```
MEMBER HOST_NAME OS_NAME OS_VERSION OS_RELEASE

0 hotel75 Linux 2 6
1 hotel78 Linux 2 6
2 hotel90 Linux 2 6
```

3 record(s) selected.

*Example 2:* Obtain memory information for the system hosting this database on a Db2 Enterprise Server Edition system:

```
select varchar(HOST_NAME,12) as HOST_NAME, MEMORY_TOTAL, MEMORY_FREE,
 MEMORY_SWAP_TOTAL, MEMORY_SWAP_FREE, VIRTUAL_MEM_TOTAL, VIRTUAL_MEM_FREE
 from table(SYSPROC.ENV_GET_SYSTEM_RESOURCES())
```

Sample output is as follows. Because of space constraints, the output is continued on a second line.

HOST_NAME	MEMORY_TOTAL	MEMORY_FREE	MEMORY_SWAP_TOTAL	...
hotel75	32189	4370	8198	...

1 record(s) selected

...	MEMORY_SWAP_FREE	VIRTUAL_MEM_TOTAL	VIRTUAL_MEM_FREE
...	7316	40387	11686

*Example 3:* Find the load times for all the systems hosting the database, in a four member partitioned database environment.

```
select MEMBER, varchar(HOST_NAME,12) as HOST_NAME,
 CPU_LOAD_SHORT, CPU_LOAD_MEDIUM, CPU_LOAD_LONG
from table(SYSPROC.ENV_GET_SYSTEM_RESOURCES()) order by MEMBER
```

Sample output is as follows:

MEMBER	HOST_NAME	CPU_LOAD_SHORT	CPU_LOAD_MEDIUM	CPU_LOAD_LONG
0	hotel75	+5.21000000000000E+000	+5.08000000000000E+000	+4.67000000000000E+000
1	hotel78	+1.33000000000000E+000	+2.18000000000000E+000	+3.66000000000000E+000
2	hotel90	+9.02000000000000E+000	+9.08000000000000E+000	+9.47000000000000E+000
3	hotel32	+1.09000000000000E+000	+1.38000000000000E+000	+1.27000000000000E+000

4 record(s) selected.

*Example 4:*

```
select MEMBER, varchar(HOST_NAME,12) as HOST_NAME, CPU_TOTAL,
 MEMORY_TOTAL, CPU_LOAD_SHORT
from table(SYSPROC.ENV_GET_SYSTEM_RESOURCES())
order by MEMBER
```

Sample output is as follows:

MEMBER	HOST_NAME	CPU_TOTAL	MEMORY_TOTAL	CPU_LOAD_SHORT
0	coralpi23	24	81920	+1.23696899414062E+000
1	coralpi23	24	81920	+1.23696899414062E+000
2	coralpi23	24	81920	+1.23696899414062E+000
3	coralpi23	24	81920	+1.23696899414062E+000

4 record(s) selected.

## ENV\_INST\_INFO administrative view - Retrieve information about the current instance

The ENV\_INST\_INFO administrative view returns information about the current instance.

The schema is SYSIBMADM.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the ENV\_INST\_INFO administrative view
- CONTROL privilege on the ENV\_INST\_INFO administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

### Example

Request information about the current instance.

```
SELECT * FROM SYSIBMADM.ENV_INST_INFO
```

The following is an example of output for this query.

```
INST_NAME IS_INST_PARTITIONABLE NUM_DBPARTITIONS INST_PTR_SIZE ...

DB2 0 1 32 ...
```

1 record(s) selected.

Output for this query (continued).

```
... RELEASE_NUM SERVICE_LEVEL BLD_LEVEL PTF FIXPACK_NUM
... -----
... 01010107 Db2 v11.1.1.1 n051106 0
```

### Information returned

Table 76. Information returned by the ENV\_INST\_INFO administrative view

Column name	Data type	Description
INST_NAME	VARCHAR(128)	Name of the current instance.
IS_INST_PARTITIONABLE	SMALLINT	Indicates whether or not the current instance is a partitionable database server instance. Possible return values are 0 if it is not a partitionable database server instance, and 1 if it is a partitionable database server instance.
NUM_DBPARTITIONS	INTEGER	Number of database partitions. If it is not a partitioned database environment, returns a value of 1.
INST_PTR_SIZE	INTEGER	Bit size of the current instance (32 or 64).
RELEASE_NUM	VARCHAR(128)	Internal release number, as returned by the db2level command; for example, 03030106.
SERVICE_LEVEL	VARCHAR(128)	Service level, as returned by the db2level command; for example, Db2 v11.1.1
BLD_LEVEL	VARCHAR(128)	Build level, as returned by the db2level command; for example, n041021.
PTF	VARCHAR(128)	Program temporary fix (PTF) identifier, as returned by the db2level command; for example, U498350.
FIXPACK_NUM	INTEGER	Fix Pack number, as returned by the db2level command; for example, 9.

Table 76. Information returned by the ENV\_INST\_INFO administrative view (continued)

Column name	Data type	Description
NUM_MEMBERS	INTEGER	The number of members on this instance.

## ENV\_PROD\_INFO administrative view - Retrieve information about installed database products

The ENV\_PROD\_INFO administrative view returns information about installed database products.

The schema is SYSIBMADM.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the ENV\_PROD\_INFO administrative view
- CONTROL privilege on the ENV\_PROD\_INFO administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

### Example

Request the installed database product information.

```
SELECT * FROM SYSIBMADM.ENV_PROD_INFO
```

The following is an example of output from this query.

```
INSTALLED_PROD INSTALLED_PROD_FULLNAME ...

ESE DB2_ENTERPRISE_SERVER_EDITION ...
WSE DB2_WORKGROUP_SERVER_EDITION ...
EXP DB2_EXPRESS_EDITION ...
```

Output from this query (continued).

```
... LICENSE_INSTALLED PROD_RELEASE LICENSE_TYPE
... -----
... Y 10.5 AUTHORIZED_USER_SINGLE_INSTALL_OPTION
... N 10.5 LICENSE_NOT_REGISTERED
... Y 10.5 RESTRICTED
```

## ENV\_PROD\_INFO administrative view metadata

Table 77. ENV\_PROD\_INFO administrative view metadata

Column name	Data type	Description
INSTALLED_PROD	VARCHAR(26)	Identifiers for database products that are installed on the system.
INSTALLED_PROD_FULLNAME	VARCHAR(100)	Full name of installed database products. Column values will be displayed in English in uppercase. Words are separated with an underscore character.

Table 77. ENV\_PROD\_INFO administrative view metadata (continued)

Column name	Data type	Description
LICENSE_INSTALLED	CHAR(1)	Indicates if product is licensed. If the value is N, the product is not licensed. If the value is Y, the product is licensed. For LICENSE_TYPE values TRIAL and LICENSE_NOT_REGISTERED, the value is always N.
PROD_RELEASE	VARCHAR(26)	Product release number.
LICENSE_TYPE	VARCHAR(50)	Name of the type of license that is installed for the product. The possible return values are: <ul style="list-style-type: none"> <li>• 12_MONTHS_LICENSE_AND_SUBSCRIPTION</li> <li>• AUTHORIZED_USER</li> <li>• AUTHORIZED_USER_SINGLE_INSTALL_OPTION</li> <li>• CLIENT_DEVICE</li> <li>• CPU</li> <li>• CPU_OPTION</li> <li>• HOST_SERVER_AND_MSU</li> <li>• LICENSE_NOT_REGISTERED</li> <li>• MANAGED_PROCESSOR</li> <li>• N/A</li> <li>• RESTRICTED</li> <li>• TRIAL</li> <li>• UNWARRANTED</li> <li>• USER</li> </ul>

## ENV\_SYS\_INFO administrative view - Retrieve information about the system

The ENV\_SYS\_INFO administrative view returns information about the system.

The schema is SYSIBMADM.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the ENV\_SYS\_INFO administrative view
- CONTROL privilege on the ENV\_SYS\_INFO administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

### Example

Request information about the system.

```
SELECT * from SYSIBMADM.ENV_SYS_INFO
```

The following is an example of output from this query.

```

OS_NAME OS_VERSION OS_RELEASE HOST_NAME

WIN32_NT 5.1 Service Pack 1 D570

1 record(s) selected.

```

Output from this query (continued).

```

... TOTAL_CPUS CONFIGURED_CPUS TOTAL_MEMORY
... -----
... 1 2 1527

```

## Information returned

Table 78. Information returned by the ENV\_SYS\_INFO administrative view

Column name	Data type	Description
OS_NAME	VARCHAR(256)	Name of the operating system.
OS_VERSION	VARCHAR(256)	Version number of the operating system.
OS_RELEASE	VARCHAR(256)	Release number of the operating system.
HOST_NAME	VARCHAR(256)	host_name - Host name monitor element
TOTAL_CPUS	INTEGER	The number of processors on this host.
CONFIGURED_CPUS	INTEGER	Number of configured physical CPUs on the system.
TOTAL_MEMORY	INTEGER	Total amount of memory on the system (in megabytes).
OS_FULL_VERSION	VARCHAR(256)	os_full_version - Operating system full version monitor element
OS_KERNEL_VERSION	VARCHAR(256)	os_kernel_version - Operating system kernel identifier monitor element
OS_ARCH_TYPE	VARCHAR(256)	os_arch_type - Operating system architecture type monitor element

## Explain routines

This grouping of routines and views can be used to retrieve information about the access plan that the optimizer chooses for an SQL statement or XQuery statement.

### EXPLAIN\_GET\_MSGS

The EXPLAIN\_GET\_MSGS table function queries the EXPLAIN\_DIAGNOSTIC and EXPLAIN\_DIAGNOSTIC\_DATA Explain tables, and returns formatted messages.

### Syntax

```

-->-- EXPLAIN_GET_MSGS—(—explain-requester—,—explain-time—,—source-name—,—>

```

```

►—source-schema—,—source-version—,—explain-level—,—stmtno—,—sectno—,—►
►—locale—)—————►

```

The schema is the same as the Explain table schema.

## Table function parameters

Any of the following input arguments can be null. If an argument is null, it is not used to limit the query.

### *explain-requester*

An input argument of type VARCHAR(128) that specifies the authorization ID of the initiator of this Explain request. A null value excludes this parameter from the search condition of the query.

### *explain-time*

An input argument of type TIMESTAMP that specifies the time of initiation for the Explain request. A null value excludes this parameter from the search condition of the query.

### *source-name*

An input argument of type VARCHAR(128) that specifies the name of the package running when the dynamic statement was explained, or the name of the source file when the static SQL statement was explained. A null value excludes this parameter from the search condition of the query.

### *source-schema*

An input argument of type VARCHAR(128) that specifies the schema, or qualifier, of the source of the Explain request. A null value excludes this parameter from the search condition of the query.

### *source-version*

An input argument of type VARCHAR(64) that specifies the version of the source of the Explain request. A null value excludes this parameter from the search condition of the query.

### *explain-level*

An input argument of type CHAR(1) that specifies the level of Explain information for which this row is relevant. A null value excludes this parameter from the search condition of the query.

### *stmtno*

An input argument of type INTEGER that specifies the statement number within the package to which this Explain information is related. A null value excludes this parameter from the search condition of the query.

### *sectno*

An input argument of type INTEGER that specifies the section number within the package to which this Explain information is related. A null value excludes this parameter from the search condition of the query.

### *locale*

An input argument of type VARCHAR(33) that specifies the locale of returned messages. If the specified locale is not installed on the database server, the value is ignored.

## Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- EXPLAIN authority

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Information returned

Table 79. Information returned by the *EXPLAIN\_GET\_MSGS* table function

Column name	Data type	Description
EXPLAIN_REQUESTER	VARCHAR(128)	Authorization ID of the initiator of this Explain request.
EXPLAIN_TIME	TIMESTAMP	Time of initiation for the Explain request.
SOURCE_NAME	VARCHAR(128)	Name of the package running when the dynamic statement was explained, or the name of the source file when the static SQL statement was explained.
SOURCE_SCHEMA	VARCHAR(128)	Schema, or qualifier, of the source of the Explain request.
SOURCE_VERSION	VARCHAR(64)	Version of the source of the Explain request.
EXPLAIN_LEVEL	CHAR(1)	Level of Explain information for which this row is relevant.
STMTNO	INTEGER	Statement number within the package to which this Explain information is related.
SECTNO	INTEGER	Section number within the package to which this Explain information is related.
DIAGNOSTIC_ID	INTEGER	ID of the diagnostic for a particular instance of a statement in the EXPLAIN_STATEMENT table.
LOCALE	VARCHAR(33)	Locale of returned messages. This locale will not match the specified locale if the latter is not installed on the database server.
MSG	VARCHAR(4096)	Formatted message text.

## Examples

Request formatted English messages from the Explain tables in the default schema for requester SIMMEN that were generated in the last hour. Specify a source name of SQLC2E03.

```
SELECT MSG
FROM TABLE(EXPLAIN_GET_MSGS(
 'SIMMEN',
 CAST(NULL AS TIMESTAMP),
 'SQLC2E03',
 CAST(NULL AS VARCHAR(128)),
 CAST(NULL AS VARCHAR(64)),
 CAST(NULL AS CHAR(1)),
 CAST(NULL AS INTEGER),
 CAST(NULL AS INTEGER),
 'en_US'))
AS REGISTRYINFO
WHERE EXPLAIN_TIME >= (CURRENT_TIMESTAMP - 1 HOUR)
ORDER BY DIAGNOSTIC_ID
```

The following is an example of output from this query.

```
MSG

EXP0012W Invalid access request. The index "index1" could not be found.
Line number "554", character number "20".
EXP0012W Invalid access request. The index "index2" could not be found.
Line number "573", character number "20".
EXP0015W Invalid join request. Join refers to tables that are not in
the same FROM clause. Line number "573", character number "20".
```

## EXPLAIN\_FORMAT\_STATS

This scalar function is used to display formatted statistics information which is parsed and extracted from explain snapshot captured for a given query.

### Syntax

►►—EXPLAIN\_FORMAT\_STATS—(—*snapshot*—)—————◄◄

The schema is SYSPROC.

### Function parameters

#### *snapshot*

An input argument of type BLOB(10M) that is the explain snapshot captured for a given query. It is stored as snapshot column of explain table *EXPLAIN\_STATEMENT*

### Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- EXPLAIN authority

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Information returned

The data type of the result is CLOB(50M).

## Example

```
SELECT EXPLAIN_FORMAT_STATS(SNAPSHOT)
FROM EXPLAIN_STATEMENT
WHERE EXPLAIN_REQUESTER = 'DB2USER1' AND
 EXPLAIN_TIME = timestamp('2006-05-12-14.38.11.109432') AND
 SOURCE_NAME = 'SQLC2F0A' AND
 SOURCE_SCHEMA = 'NULLID' AND
 SOURCE_VERSION = '' AND
 EXPLAIN_LEVEL = '0' AND
 STMTNO = 1 AND
 SECTNO = 201
```

The following is a sample output of this function:

Tablespace Context:

```

Name: USERSPACE1
Overhead: 7.500000
Transfer Rate: 0.060000
Prefetch Size: 32
Extent Size: 32
Type: Database managed
Partition Group Name: NULLP
Buffer Pool Identifier: 0
```

Base Table Statistics:

```

Name: T1
Schema: DB2USER2
Number of Columns: 3
Number of Pages with Rows: 1
Number of Pages: 1
Number of Rows: 5
Table Overflow Record Count: 0
Width of Rows: 26
Time of Creation: 2006-06-16-11.46.53.041085
Last Statistics Update: 2006-06-26-12.23.44.814201
Statistics Type: Fabrication
Primary Tablespace: USERSPACE1
Tablespace for Indexes: USERSPACE1
Tablespace for Long Data: NULLP
Number of Referenced Columns: 2
Number of Indexes: 1
Volatile Table: No
Table Active Blocks: 1
Number of Column Groups: 0
Number of Data Partitions: 1
Average Row Compression Ratio: -9.000000
Percent Rows Compressed: -9.000000
Average Compressed Row Size: -9
Statistics Type: U
```

Column Information:

```

Number: 1
Name: C1
```

Statistics Available: Yes

Column Statistics:

-----  
Schema name of the column type: SYSIBM  
Name of column type: INTEGER  
Maximum column length: 4  
Scale for decimal column: 0  
Number of distinct column values: 4  
Average column length: 5  
Number of most frequent values: 1  
Number of quantiles: 5  
Second highest data value: 3  
Second lowest data value: 2  
Column sequence in partition key: 0  
Average number of sub-elements: -1  
Average length of delimiters: -1

Column Distribution Statistics:

Frequency Statistics:

Valcount	Value
2	1

Quantile Statistics:

Valcount	Distcount	Value
0	1	1
2	1	1
3	2	2
4	3	3
5	4	4

Column Information:

-----  
Number: 2  
Name: C2  
Statistics Available: Yes

Column Statistics:

-----  
Schema name of the column type: SYSIBM  
Name of column type: INTEGER  
Maximum column length: 4  
Scale for decimal column: 0  
Number of distinct column values: 4  
Average column length: 5  
Number of most frequent values: 1  
Number of quantiles: 5  
Second highest data value: 3  
Second lowest data value: 2  
Column sequence in partition key: 0  
Average number of sub-elements: -1  
Average length of delimiters: -1

Column Distribution Statistics:

Frequency Statistics:

Valcount	Value
2	1

Quantile Statistics:

Valcount	Distcount	Value
0	0	1

2	0	1
3	0	2
4	0	4
5	0	4

#### Indexes defined on the table:

```

Name: IDX_T1C1C2
Schema: DB2USER2
Unique Rule: Duplicate index
Used in Operator: Yes
Page Fetch Pairs: Not Available
Number of Columns: 2
Index Leaf Pages: 1
Index Tree Levels: 1
Index First Key Cardinality: 4
Index Full Key Cardinality: 4
Index Cluster Ratio: 100
Index Cluster Factor: -1.000000
Time of Creation: 2006-06-16-11.46.53.596717
Last Statistics Update: 2006-06-26-12.23.44.814201
Index Sequential Pages: 0
Index First 2 Keys Cardinality: 4
Index First 3 Keys Cardinality: -1
Index First 4 Keys Cardinality: -1
Index Avg Gap between Sequences: 0.000000
Fetch Avg Gap between Sequences: -1.000000
Index Avg Sequential Pages: 0.000000
Fetch Avg Sequential Pages: -1.000000
Index Avg Random Pages: 1.000000
Fetch Avg Random Pages: -1.000000
Index RID Count: 5
Index Deleted RID Count: 0
Index Empty Leaf Pages: 0
Avg Partition Cluster Ratio: -1
Avg Partition Cluster Factor: -1.000000
Data Partition Cluster Factor: 1.000000
Data Partition Page Fetch Pairs: Not Available

```

#### Base Table Statistics:

```

Name: T2
Schema: DB2USER2
Number of Columns: 3
Number of Pages with Rows: 1
Number of Pages: 1
Number of Rows: 2
Table Overflow Record Count: 0
Width of Rows: 26
Time of Creation: 2006-06-16-11.46.53.398092
Last Statistics Update: 2006-06-26-12.23.45.157028
Statistics Type: Synchronous
Primary Tablespace: USERSPACE1
Tablespace for Indexes: USERSPACE1
Tablespace for Long Data: NULLP
Number of Referenced Columns: 2
Number of Indexes: 1
Volatile Table: No
Table Active Blocks: -1
Number of Column Groups: 0
Number of Data Partitions: 1

```

#### Column Information:

```

Number: 1
Name: C1
Statistics Available: Yes

```

Column Statistics:

```

Schema name of the column type: SYSIBM
Name of column type: INTEGER
Maximum column length: 4
Scale for decimal column: 0
Number of distinct column values: 2
Average column length: 5
Number of most frequent values: -1
Number of quantiles: 2
Second highest data value: 2
Second lowest data value: 1
Column sequence in partition key: 0
Average number of sub-elements: -1
Average length of delimiters: -1

```

Column Distribution Statistics:

Quantile Statistics:

Valcount	Distcount	Value
1	1	1
2	2	2

Column Information:

```

Number: 2
Name: C2
Statistics Available: Yes

```

Column Statistics:

```

Schema name of the column type: SYSIBM
Name of column type: INTEGER
Maximum column length: 4
Scale for decimal column: 0
Number of distinct column values: 2
Average column length: 5
Number of most frequent values: -1
Number of quantiles: 2
Second highest data value: 2
Second lowest data value: 1
Column sequence in partition key: 0
Average number of sub-elements: -1
Average length of delimiters: -1

```

Column Distribution Statistics:

Quantile Statistics:

Valcount	Distcount	Value
1	0	1
2	0	2

Indexes defined on the table:

```

Name : IDX_T2C1
Schema: DB2USER2
Unique Rule: Duplicate index
Used in Operator: No
Page Fetch Pairs: Not Available
Number of Columns: 1
Index Leaf Pages: 1
Index Tree Levels: 1
Index First Key Cardinality: 2
Index Full Key Cardinality: 2

```

Index Cluster Ratio:	100
Index Cluster Factor:	-1.000000
Time of Creation:	2006-06-16-11.46.53.857520
Last Statistics Update:	2006-06-26-12.23.45.157028
Index Sequential Pages:	0
Index First 2 Keys Cardinality:	-1
Index First 3 Keys Cardinality:	-1
Index First 4 Keys Cardinality:	-1
Index Avg Gap between Sequences:	0.000000
Fetch Avg Gap between Sequences:	-1.000000
Index Avg Sequential Pages:	0.000000
Fetch Avg Sequential Pages:	-1.000000
Index Avg Random Pages:	1.000000
Fetch Avg Random Pages:	-1.000000
Index RID Count:	2
Index Deleted RID Count:	0
Index Empty Leaf Pages:	0
Avg Partition Cluster Ratio:	-1
Avg Partition Cluster Factor:	-1.000000
Data Partition Cluster Factor:	1.000000
Data Partition Page Fetch Pairs:	Not Available

## EXPLAIN\_FROM\_ACTIVITY procedure - Explain statement using activity event monitor information

The EXPLAIN\_FROM\_ACTIVITY procedure explains a specific execution of a statement using the contents of the section obtained from an activity event monitor.

The Explain output is placed in the Explain tables for processing using any existing Explain tools (for example, **db2exfmt**). The Explain output contains, if available, both the access plan and section actuals (runtime statistics for operators in the access plan).

```

▶▶—EXPLAIN_FROM_ACTIVITY—————▶
▶—(—appl_id—,—uow_id—,—activity_id—,—activity_evmon_name—,—explain_schema—————▶
▶,—explain_requester—,—explain_time—,—source_name—,—source_schema—,—source_version—)————▶▶

```

The schema is SYSPROC.

## Authorization

One of the following authorities or privileges is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- EXPLAIN authority

In addition, the privileges held by the authorization ID of the session, including privileges granted to groups, must include at least one of the following:

- INSERT privilege on the explain tables in the specified schema
- CONTROL privilege on the explain tables in the specified schema
- DATAACCESS authority

## Default PUBLIC privilege

None

## Table function parameters

### ***appl\_id***

An input argument of type VARCHAR(64) that uniquely identifies the application that issued the activity whose section is to be explained. If *appl\_id* is null or an empty string, SQL2032N is returned.

### ***uow\_id***

An input argument of type INTEGER specifying the unit of work identifier for the activity whose section is to be explained. Unit of work ID is only unique within a given application. If *uow\_id* is null, SQL2032N is returned.

### ***activity\_id***

An input argument of type INTEGER specifying the identifier of the activity whose section is to be explained. Activity ID is only unique within a unit of work. If *activity\_id* is null, SQL2032N is returned.

### ***activity\_evmon\_name***

An input argument of VARCHAR(128) that specifies the name of a write to table activity event monitor containing the activity whose section is to be explained. If the event monitor does not exist or is not an activity event monitor, SQL0204N is returned. If the event monitor is not a write to table event monitor, SQL20502N is returned. If *activity\_evmon\_name* is not specified, SQL2032N is returned. If the caller does not have SELECT privilege on the activity event monitor tables, SQL0551N is returned.

### ***explain\_schema***

An optional input or output argument of type VARCHAR(128) that specifies the schema containing the Explain tables where the explain information should be written. If an empty string or NULL is specified, a search is made for the explain tables under the session authorization ID and, following that, the SYSTOOLS schema. If the Explain tables cannot be found, SQL0219N is returned. If the caller does not have INSERT privilege on the Explain tables, SQL0551N is returned. On output, this parameter is set to the schema containing the Explain tables where the information was written.

### ***explain\_requester***

An output argument of type VARCHAR(128) that contains the session authorization ID of the connection in which this routine was invoked.

### ***explain\_time***

An output argument of type TIMESTAMP that contains the time of initiation for the Explain request.

### ***source\_name***

An output argument of type VARCHAR(128) that contains the name of the package running when the statement was prepared or compiled.

### ***source\_schema***

An output argument of type VARCHAR(128) that contains the schema, or qualifier, of the source of Explain request.

### ***source\_version***

An output argument of type VARCHAR(64) that contains the version of the source of the Explain request.

## Example

The following example assumes that you are mining the data collected in the activity event monitor over a period of time and using the following query, you notice a particularly expensive SQL statement in terms of CPU cost.

```
SELECT APPL_ID,
 UOW_ID,
 ACTIVITY_ID,
 USER_CPU_TIME
FROM ACTIVITY_A
ORDER BY USER_CPU_TIME
```

The following example shows output from this query. The application with an ID of N2.DB2INST1.0B5A12222841 is using a large amount of CPU time.

APPL_ID	UOW_ID	ACTIVITY_ID	USER_CPU_TIME
*N2.DB2INST1.0B5A12222841	1	1	92782334234
*N2.DB2INST1.0B5A12725841	2	7	326

2 record(s) selected.

You can use the EXPLAIN\_FROM\_ACTIVITY procedure to investigate the access plan for this activity, to determine if the activity could benefit from tuning, for example, by adding an index.

```
CALL EXPLAIN_FROM_ACTIVITY('*N2.DB2INST1.0B5A12222841', 1, 1, 'A', 'MYSHEMA',
 ?, ?, ?, ?, ?)
```

## Usage notes

In order to run Explain on the section of the activity, you must specify the COLLECT ACTIVITY DATA WITH SECTION clause when you enable collection of activity data, so that the section is collected with the activity information. If there is no section stored with the identified activity entry, SQL20501 is returned.

If section actuals were not collected for an activity, the section explain will succeed, but the Explain output will not include actuals information. Section actuals are not collected in the following cases:

- The activity specified as input was captured using the WLM\_CAPTURE\_ACTIVITY\_IN\_PROGRESS stored procedure. In this case, the value of the *partial\_record* element in the activity logical group is 1.
- The activity event monitor ACTIVITY table is missing the SECTION\_ACTUALS element.
- The section executed is a static section and it has not been rebound since applying Db2 Version 9.7 Fix Pack 1.
- Section actuals were not enabled for the section that was captured. Section actuals are enabled using the **section\_actuals** database configuration parameter or for a specific application using the **WLM\_SET\_CONN\_ENV** procedure. By default, section actuals are disabled.

**Note:** To verify that section actuals were collected for an activity, check whether the SECTION\_ACTUALS element in the ACTIVITY table has a length greater than 0.

**Note:** The **section\_actuals** setting specified by the **WLM\_SET\_CONN\_ENV** procedure for an application takes effect immediately. Section actuals will be collected for the next statement issued by the application.

**Note:** In a partitioned database environment, section actuals will be collected only on members where activity data is collected. To collect actuals on all members, ensure the activity is collected using the COLLECT ACTIVITY DATA ON ALL MEMBERS WITH DETAILS, SECTION clause. If the user wants to enable collection at all members for a particular application, they can include the *<collectactpartition>* tag with a value of "ALL" in the second argument when calling the WLM\_SET\_CONN\_ENV procedure.

If no activity can be found that corresponds to the *appl\_id*, *uow\_id*, and *activity\_id* that you input, SQL20501 is returned. If multiple activities match, which may occur if an activity was collected multiple times during execution using the WLM\_CAPTURE\_ACTIVITY\_IN\_PROGRESS stored procedure, the most recent entry for which a section was captured will be used for Explain.

The output parameters *explain\_requester*, *explain\_time*, *source\_name*, *source\_schema*, and *source\_version* comprise the key used to look up the Explain information for the section in the Explain tables. Use these parameters with any existing Explain tools (for example, **db2exfmt**) to format the explain information retrieved from the section.

The EXPLAIN\_FROM\_ACTIVITY procedure does not issue a COMMIT statement after inserting into the Explain tables. It is the responsibility of the caller of the procedure to issue a COMMIT.

The following elements must be present in the ACTIVITYSTMT logical group: STMT\_TEXT, ORIGINAL\_STMT\_TEXT, SECTION\_ENV, EXECUTABLE\_ID, APPL\_ID, ACTIVITY\_ID, UOW\_ID. If any of these elements are missing, the stored procedure returns SQL206.

### **EXPLAIN\_FROM\_CATALOG procedure - Explain a statement using section information from catalogs**

The EXPLAIN\_FROM\_CATALOG procedure explains a statement using the contents of the section obtained from the catalogs.

The Explain output is placed in the Explain tables for processing using any existing explain tools (for example, **db2exfmt**).

### **Authorization**

One of the following authorities or privileges is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- EXPLAIN authority

In addition, the privileges held by the authorization ID of the session, including privileges granted to groups, must include at least one of the following:

- INSERT privilege on the explain tables in the specified schema
- CONTROL privilege on the explain tables in the specified schema
- DATAACCESS authority

## Default PUBLIC privilege

None

## Syntax

```
►—EXPLAIN_FROM_CATALOG—(—pkgschema—,—pkgname—,—pkgversion—,—sectno—,—explain_schema—►
►,—explain_requester—,—explain_time—,—source_name—,—source_schema—,—source_version—)►◄
```

The schema is SYSPROC.

## Procedure parameters

### ***pkgschema***

An input argument of type VARCHAR(128) specifying the schema of the package containing the section to be explained. If *pkgschema* is null or an empty string, SQL2032N is returned.

### ***pkgname***

An input argument of type VARCHAR(128) specifying the package containing the section to be explained. If *pkgname* is null or an empty string, SQL2032N is returned.

### ***pkgversion***

An input argument of type VARCHAR(64) specifying the version identifier for the package containing the section to be explained. Specify an empty string if the package has no version (a blank ' ' character if VARCHAR2 compatibility mode is enabled). If *pkgversion* is null, SQL2032N is returned.

### ***sectno***

An input argument of type SMALLINT specifying the section to be explained. If *sectno* is null, SQL2032N is returned.

### ***explain\_schema***

An optional input or output argument of type VARCHAR(128) that specifies the schema containing the Explain tables where the explain information should be written. If an empty string or NULL is specified, a search is made for the explain tables under the session authorization ID and, following that, the SYSTOOLS schema. If the Explain tables cannot be found, SQL0219N is returned. If the caller does not have INSERT privilege on the Explain tables, SQL0551N is returned. On output, this parameter is set to the schema containing the Explain tables where the information was written.

### ***explain\_requester***

An output argument of type VARCHAR(128) that contains the session authorization ID of the connection in which this routine was invoked.

### ***explain\_time***

An output argument of type TIMESTAMP that contains the time of initiation for the Explain request.

### ***source\_name***

An output argument of type VARCHAR(128) that contains the name of the package running when the statement was prepared or compiled.

### ***source\_schema***

An output argument of type VARCHAR(128) that contains the schema, or qualifier, of the source of Explain request.

### ***source\_version***

An output argument of type VARCHAR(64) that contains the version of the source of the Explain request.

### **Usage notes**

If no section can be found corresponding to the input parameters, SQL20501 is returned.

The output parameters *explain\_requester*, *explain\_time*, *source\_name*, *source\_schema*, *source\_version* comprise the key used to look up the Explain information for the section in the Explain tables. Use these parameters with any existing Explain tools (for example, **db2exfmt**) to format the explain information retrieved from the section.

The procedure does not issue a COMMIT statement after inserting into the Explain tables. It is the responsibility of the caller of the procedure to issue a COMMIT.

### **Example**

The following example demonstrates how to explain a static statement that was compiled and exists in the catalogs. First, you can identify the section by selecting from the SYSCAT.STATEMENTS catalog view, for example:

```
SELECT pkgschema,
 pkgname,
 version,
 Sectno
FROM SYSCAT.STATEMENTS
WHERE TEXT = 'select count(*) from syscat.tables'
```

This query returns the following sample output:

PKGSHEMA	PKGNAME	VERSION	SECTNO
NULLID	SQL2G0S		1
NULLID	SQL2G0S	VERSION1	1

2 record(s) selected.

Then pass the *pkgschema*, *pkgname*, *version* and *sectno* identification information into the EXPLAIN\_FROM\_CATALOG procedure, for example:

```
CALL EXPLAIN_FROM_CATALOG('NULLID', 'SQL2G0S', '', 1, 'MYSHEMA', ?, ?, ?, ?, ?)
```

### **EXPLAIN\_FROM\_DATA procedure - Explain a statement using the input section**

The EXPLAIN\_FROM\_DATA procedure explains a statement using the contents of the input section.

The Explain output is placed in the Explain tables for processing using any existing Explain tools (for example, **db2exfmt**).

### **Authorization**

One of the following authorities or privileges is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority

- SQLADM authority
- EXPLAIN authority

In addition, the privileges held by the authorization ID of the session, including privileges granted to groups, must include at least one of the following:

- INSERT privilege on the explain tables in the specified schema
- CONTROL privilege on the explain tables in the specified schema
- DATAACCESS authority

## Default PUBLIC privilege

None

## Syntax

```
►►EXPLAIN_FROM_DATA(—section—,—stmt_text—,—executable_id—,—explain_schema—►►
►,—explain_requester—,—explain_time—,—source_name—,—source_schema—,—source_version—)►►
```

The schema is SYSPROC.

## Procedure parameters

### *section*

An input argument of type BLOB(134M) that contains the section to be explained. You can obtain the section from various sources, including event monitor tables and the catalog tables. SQL20503N is returned in the following situations:

- The input section is not a valid section
- The input section is from a database with a different operating system

### *stmt\_text*

An optional input argument of type CLOB(2M) that contains the text of the statement corresponding to the input section. If *stmt\_text* is NULL, the formatted Explain output will not contain any statement text.

### *executable\_id*

An optional input argument of type VARCHAR(32) FOR BIT DATA that contains the executable ID used to identify the section. If *executable\_id* is NULL, the formatted explain output will not contain an executable ID.

### *explain\_schema*

An optional input or output argument of type VARCHAR(128) that specifies the schema containing the Explain tables where the explain information should be written. If an empty string or NULL is specified, a search is made for the explain tables under the session authorization ID and, following that, the SYSTOOLS schema. If the Explain tables cannot be found, SQL0219N is returned. If the caller does not have INSERT privilege on the Explain tables, SQL0551N is returned. On output, this parameter is set to the schema containing the Explain tables where the information was written.

### *explain\_requester*

An output argument of type VARCHAR(128) that contains the session authorization ID of the connection in which this routine was invoked.

*explain\_time*

An output argument of type `TIMESTAMP` that contains the time of initiation for the Explain request.

***source\_name***

An output argument of type VARCHAR(128) that contains the name of the package running when the statement was prepared or compiled.

***source\_schema***

An output argument of type VARCHAR(128) that contains the schema, or qualifier, of the source of Explain request.

***source\_version***

An output argument of type VARCHAR(64) that contains the version of the source of the Explain request.

## Restrictions

The section that is passed as input to the EXPLAIN\_FROM\_DATA stored procedure must be obtained from a database whose architecture is the same as the database used for the EXPLAIN\_FROM\_DATA stored procedure call.

## Usage notes

The input section can be obtained from a number of different sources:

- Activity event monitor
- Package cache event monitor
- Catalog tables
- Any user table or input source that has made a copy of the section from one of the locations listed previously.

The output parameters *explain\_requester*, *explain\_time*, *source\_name*, *source\_schema*, *source\_version* comprise the key used to look up the Explain information for the section in the Explain tables. Use these parameters with any existing Explain tools (for example, **db2exfmt**) to format the explain information retrieved from the section.

The procedure does not issue a COMMIT after inserting into the Explain tables. It is the responsibility of the caller of the procedure to issue a COMMIT.

### Example

[illegible]

You issue the `EXPLAIN_FROM_DATA` procedure to understand the access plan for this statement, passing as input the section from the entry in the `PKGCACHE` table. You place the Explain output in the explain tables in the `MYSCHEMA` schema.

```
SET SERVEROUTPUT ON;
```

BEGIN

```
DECLARE EXECUTABLE ID VARCHAR(32) FOR BIT DATA; --
```



In addition, the privileges held by the authorization ID of the session, including privileges granted to groups, must include at least one of the following:

- INSERT privilege on the explain tables in the specified schema
- CONTROL privilege on the explain tables in the specified schema
- DATAACCESS authority

## Default PUBLIC privilege

None

## Syntax

```
►►—EXPLAIN_FROM_SECTION—►►
►—(—executable_id—,—section_source_type—,—section_source_name—,—member—,—explain_schema—►
►,—,—explain_requester—,—explain_time—,—source_name—,—source_schema—,—source_version—)►►
```

## Procedure parameters

### ***executable\_id***

An input argument of type VARCHAR(32) FOR BIT DATA that uniquely identifies a section to be explained. If this argument is null or an empty string, SQL2032 is returned.

### ***section\_source\_type***

An input argument of type CHAR(1) that specifies the source of the section to be explained. Valid values are:

- M - Section is obtained from the in-memory package cache
- P - Section is obtained from a package cache event monitor. Either regular or unformatted event tables can be used as the output type for the event monitor.

For static SQL, if the *section\_source\_type* is M and the section cannot be located in the package cache, the catalog tables are searched for the section.

### ***section\_source\_name***

An input argument of VARCHAR(128) that specifies the name of a package cache event monitor if the *section\_source\_type* is P. If the *section\_source\_type* is M, the name of a package cache event monitor can be optionally specified. The event monitor is searched for the section if the section cannot be found in the package cache (for example, if the section was flushed from the package cache before the EXPLAIN\_FROM\_SECTION stored procedure was invoked). If the source input event monitor is not a package cache event monitor created with the COLLECT DETAILED DATA option, SQL0204N is returned. If the caller does not have SELECT privilege on the package cache event monitor table, SQL0551N is returned.

### ***member***

An input argument of type INTEGER that specifies the member where the section to be explained resides in memory if the *section\_source\_type* is M. If -1 is specified, the procedure searches for the section on the current coordinator member and the section compilation member. This argument is ignored if the *section\_source\_type* is anything other than M.

### ***explain\_schema***

An optional input or output argument of type VARCHAR(128) that specifies the schema containing the Explain tables where the explain information should

be written. If an empty string or NULL is specified, a search is made for the explain tables under the session authorization ID and, following that, the SYSTOOLS schema. If the Explain tables cannot be found, SQL0219N is returned. If the caller does not have INSERT privilege on the Explain tables, SQL0551N is returned. On output, this parameter is set to the schema containing the Explain tables where the information was written.

***explain\_requester***

An output argument of type VARCHAR(128) that contains the session authorization ID of the connection in which this routine was invoked.

***explain\_time***

An output argument of type TIMESTAMP that contains the time of initiation for the Explain request.

***source\_name***

An output argument of type VARCHAR(128) that contains the name of the package running when the statement was prepared or compiled.

***source\_schema***

An output argument of type VARCHAR(128) that contains the schema, or qualifier, of the source of Explain request.

***source\_version***

An output argument of type VARCHAR(64) that contains the version of the source of the Explain request.

## Usage notes

If the section corresponding to the input executable ID cannot be found, SQL20501 is returned. The input *executable\_id* can be obtained from the following sources

- Activity event monitor
- Package cache event monitor
- MON\_GET\_ACTIVITY\_DETAILS table function
- MON\_GET\_PKG\_CACHE\_STMT table function
- WLM\_GET\_WORKLOAD\_OCCURRENCE\_ACTIVITIES table function
- WLM\_GET\_SERVICE\_CLASS\_AGENTS table function
- MON\_GET\_PKG\_CACHE\_STMT\_DETAILS table function
- MON\_GET\_APPL\_LOCKWAIT table function

The output parameters *explain\_requester*, *explain\_time*, *source\_name*, *source\_schema*, *source\_version* comprise the key used to look up the information for the section in the Explain tables. Use these parameters with any existing Explain tools (for example, **db2exfmt**) to format the explain information retrieved from the section.

The procedure does not issue a COMMIT statement after inserting into the Explain tables. It is the responsibility of the caller of the procedure to issue a COMMIT.

## Example

This example shows how to identify and analyze a particularly expensive statement in the package cache by looking at the monitoring metrics available per section. First, issue a query similar to the following SELECT statement to determine the CPU time usage of sections.

```

SELECT executable_id,
 Total_cpu_time,
 Varchar(stmt_text, 100) as stmt_text
FROM TABLE(MON_GET_PKG_CACHE_STMT (NULL, NULL,NULL, -1)) AS T

```

The following sample shows output from this query.

```

EXECUTABLE_ID TOTAL_CPU_TIME ...

x'010000000000000012...200200811261904103698' 91875622 ...
x'010000000000000007...200200811261904103238' 300 ...

```

2 record(s) selected.

The following sample continues the output from this query.

```

...STMT_TEXT
...-----
...SELECT * FROM SYSCAT.TABLES
...INSERT INTO T1 VALUES(123)

```

2 record(s) selected.

To examine the access plan for the expensive SELECT statement, pass its *executable\_id* to the EXPLAIN\_FROM\_SECTION procedure. Place the output in the Explain tables in the MYSCHEMA schema.

```

CALL EXPLAIN_FROM_SECTION
(x'0100000000000000120000000000000000000000000000000000200200811261904103698',
 'M', NULL, 0, 'MYSCHEMA', ?, ?, ?, ?, ?)

```

## Monitor views

Monitor views are an ideal place to start when interested in basic monitoring.

These views provide a condensed representation of monitoring information, displaying the highlights, as well as providing some of the standard calculations of most interest, such as buffer pool hit ratio.

Views are very similar to table functions; they return data in table format, but unlike table functions, they do not require any input parameters.

The monitor views return metrics on various database activities, for example:

- MON\_BP\_UTILIZATION returns key monitoring metrics, including hit ratios and average read and write times, for all buffer pools and all database partitions in the currently connected database.
- MON\_CURRENT\_SQL returns metrics for all activities that were submitted on all members of the database and have not yet been completed, including a point-in-time view of currently executing SQL statements.
- MON\_DB\_SUMMARY returns metrics aggregated over all service classes.
- MON\_LOCKWAITS returns information about agents working on behalf of applications that are waiting to obtain locks in the currently connected database.
- MON\_SERVICE\_SUBCLASS\_SUMMARY returns metrics for all service subclasses, showing work executed per service class.
- MON\_CURRENT\_UOW returns metrics for all units of work.
- MON\_WORKLOAD\_SUMMARY returns metrics for all workloads, showing incoming work per workload.

## MON\_BP\_UTILIZATION administrative view - Retrieve metrics for bufferpools

The MON\_BP\_UTILIZATION administrative view returns key monitoring metrics, including hit ratios and average read and write times, for all buffer pools and all database partitions in the currently connected database.

It provides information that is critical for performance monitoring, because it helps you check how efficiently you are using your buffer pools.

The schema is SYSIBMADM.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the MON\_BP\_UTILIZATION administrative view
- CONTROL privilege on the MON\_BP\_UTILIZATION administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Information returned

Table 80. Information returned by the MON\_BP\_UTILIZATION administrative view

Column name	Data type	Description or Monitor element
BP_NAME	VARCHAR(128)	bp_name - Buffer pool name
MEMBER	SMALLINT	member - Database member
DATA_PHYSICAL_READS	BIGINT	Indicates the number of data pages read from the table space containers (physical) for temporary as well as regular and large table spaces. This is calculated as ( <i>pool_data_p_reads</i> + <i>pool_temp_data_p_reads</i> ) where <i>pool_data_p_reads</i> and <i>pool_temp_data_p_reads</i> represent the following monitor elements: <ul style="list-style-type: none"><li>• <i>pool_data_p_reads</i> - Buffer pool data physical reads</li><li>• <i>pool_temp_data_p_reads</i> - Buffer pool temporary data physical reads</li></ul>
DATA_HIT_RATIO_PERCENT	DECIMAL(5,2)	Data hit ratio, that is, the percentage of time that the database manager did not need to load a page from disk to service a data page request. In Db2 pureScale environments, this value is the percentage of time that the database manager located a data page in the local buffer pool.

Table 80. Information returned by the MON\_BP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
INDEX_PHYSICAL_READS	BIGINT	Indicates the number of index pages read from the table space containers (physical) for temporary as well as regular and large table spaces. This is calculated as ( <i>pool_index_p_reads</i> + <i>pool_temp_index_p_reads</i> ) where <i>pool_index_p_reads</i> + <i>pool_temp_index_p_reads</i> represent the following monitor elements: <ul style="list-style-type: none"> <li>• <i>pool_index_p_reads</i> - Buffer pool index physical reads</li> <li>• <i>pool_temp_index_p_reads</i> - Buffer pool temporary index physical reads</li> </ul>
INDEX_HIT_RATIO_PERCENT	DECIMAL(5,2)	Index hit ratio, that is, the percentage of time that the database manager did not need to load a page from disk to service an index data page request. In Db2 pureScale environments, this value is the percentage of time that the database manager located a data page in the local buffer pool.
XDA_PHYSICAL_READS	BIGINT	Indicates the number of data pages for XML storage objects (XDAs) read from the table space containers (physical) for temporary as well as regular and large table spaces. This is calculated as ( <i>pool_xda_p_reads</i> + <i>pool_temp_xda_p_reads</i> ) where <i>pool_xda_p_reads</i> and <i>pool_temp_xda_p_reads</i> represent the following monitor elements: <ul style="list-style-type: none"> <li>• <i>pool_xda_p_reads</i> - Buffer pool XDA data physical reads</li> <li>• <i>pool_temp_xda_p_reads</i> - Buffer pool temporary XDA data physical reads</li> </ul>
XDA_HIT_RATIO_PERCENT	DECIMAL(5,2)	Auxiliary storage objects hit ratio, that is, the percentage of time that the database manager did not need to load a page from disk to service a data page request for XML storage objects (XDAs). In a Db2 pureScale environment, this value is the percentage of time the database manager used to locate a data page for an XDA in the local buffer pool.

Table 80. Information returned by the MON\_BP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
COL_PHYSICAL_READS	BIGINT	<p>Indicates the number of column-organized table data pages read from the physical table space containers for temporary, regular, and large table spaces. This is calculated as (<i>pool_col_p_reads</i> + <i>pool_temp_col_p_reads</i>) where <i>pool_col_p_reads</i> and <i>pool_temp_col_p_reads</i> represent the following monitor elements:</p> <ul style="list-style-type: none"> <li>• <i>pool_col_p_reads</i> - Buffer pool column-organized physical reads</li> <li>• <i>pool_temp_col_p_reads</i> - Buffer pool column-organized temporary physical reads</li> </ul>
COL_HIT_RATIO_PERCENT	DECIMAL(5,2)	<p>Column-organized table data hit ratio. The percentage of time that the database manager did not need to load a page from disk to service a column-organized table data page request. In Db2 pureScale environments, this value is the percentage of time that the database manager located a data page in the local buffer pool.</p>

Table 80. Information returned by the MON\_BP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
TOTAL_PHYSICAL_READS	BIGINT	<p>Indicates the number of data pages, index pages, and data pages for XML storage objects (XDAs) read from the table space containers (physical) for temporary as well as regular and large table spaces.</p> <p>This is calculated as  <math>(pool\_data\_p\_reads + pool\_temp\_data\_p\_reads + pool\_index\_p\_reads + pool\_temp\_index\_p\_reads + pool\_xda\_p\_reads + pool\_temp\_xda\_p\_reads + pool\_col\_p\_reads + pool\_temp\_col\_p\_reads)</math> where  <math>pool\_data\_p\_reads</math>,  <math>pool\_temp\_data\_p\_reads</math>,  <math>pool\_index\_p\_reads</math>,  <math>pool\_temp\_index\_p\_reads</math>,  <math>pool\_xda\_p\_reads</math>,  <math>pool\_temp\_xda\_p\_reads</math>,  <math>pool\_col\_p\_reads</math>, and  <math>pool\_temp\_col\_p\_reads</math> represent the following monitor elements:</p> <ul style="list-style-type: none"> <li>• <math>pool\_data\_p\_reads</math> - Buffer pool data physical reads</li> <li>• <math>pool\_temp\_data\_p\_reads</math> - Buffer pool temporary data physical reads</li> <li>• <math>pool\_index\_p\_reads</math> - Buffer pool index physical reads</li> <li>• <math>pool\_temp\_index\_p\_reads</math> - Buffer pool temporary index physical reads</li> <li>• <math>pool\_xda\_p\_reads</math> - Buffer pool XDA data physical reads</li> <li>• <math>pool\_temp\_xda\_p\_reads</math> - Buffer pool temporary XDA data physical reads</li> <li>• <math>pool\_col\_p\_reads</math> - Buffer pool column-organized physical reads</li> <li>• <math>pool\_temp\_col\_p\_reads</math> - Buffer pool column-organized temporary physical reads</li> </ul>

Table 80. Information returned by the MON\_BP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
AVG_PHYSICAL_READ_TIME	BIGINT	<p>Average time, in milliseconds, spent reading pages from the table space containers (physical) for all types of table spaces.</p> <p>If the sum of physical reads is greater than zero, this is calculated as <math>\text{pool\_read\_time} / (\text{pool\_data\_p\_reads} + \text{pool\_temp\_data\_p\_reads} + \text{pool\_index\_p\_reads} + \text{pool\_temp\_index\_p\_reads} + \text{pool\_xda\_p\_reads} + \text{pool\_temp\_xda\_p\_reads} + \text{pool\_col\_p\_reads} + \text{pool\_temp\_col\_p\_reads})</math> where <i>pool_read_time</i>, <i>pool_data_p_reads</i>, <i>pool_temp_data_p_reads</i>, <i>pool_index_p_reads</i>, <i>pool_temp_index_p_reads</i>, <i>pool_xda_p_reads</i>, <i>pool_temp_xda_p_reads</i>, <i>pool_col_p_reads</i>, and <i>pool_temp_col_p_reads</i> represent the following monitor elements:</p> <ul style="list-style-type: none"> <li>• <i>pool_read_time</i> - Total buffer pool physical read time</li> <li>• <i>pool_data_p_reads</i> - Buffer pool data physical reads</li> <li>• <i>pool_temp_data_p_reads</i> - Buffer pool temporary data physical reads</li> <li>• <i>pool_index_p_reads</i> - Buffer pool index physical reads</li> <li>• <i>pool_temp_index_p_reads</i> - Buffer pool temporary index physical reads</li> <li>• <i>pool_xda_p_reads</i> - Buffer pool XDA data physical reads</li> <li>• <i>pool_temp_xda_p_reads</i> - Buffer pool temporary XDA data physical reads</li> <li>• <i>pool_col_p_reads</i> - Buffer pool column-organized physical reads</li> <li>• <i>pool_temp_col_p_reads</i> - Buffer pool column-organized temporary physical reads</li> </ul> <p>If the sum of physical reads is not greater than zero, NULL is returned.</p>

Table 80. Information returned by the MON\_BP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
PREFETCH_RATIO_PERCENT	DECIMAL(5,2)	Percentage of pages read asynchronously (with prefetching). If many applications are reading data synchronously without prefetching, your system might not be tuned optimally.
ASYNC_NOT_READ_PERCENT	DECIMAL(5,2)	<p>Percentage of pages read asynchronously from disk, but never accessed by a query. If too many pages are read asynchronously from disk into the bufferpool, but no query ever accesses those pages, the prefetching might degrade performance.</p> <p>If the sum of asynchronous reads is greater than zero, this is calculated as <math>\text{unread\_prefetch\_pages} / (\text{pool\_async\_data\_reads} + \text{pool\_async\_index\_reads} + \text{pool\_async\_xda\_reads} + \text{pool\_async\_col\_reads})</math> where <i>unread_prefetch_pages</i>, <i>pool_async_data_reads</i>, <i>pool_async_index_reads</i>, <i>pool_async_xda_reads</i>, and <i>pool_async_col_reads</i> represent the following monitor elements:</p> <ul style="list-style-type: none"> <li>• unread_prefetch_pages - Unread prefetch pages</li> <li>• pool_async_data_reads - Buffer pool asynchronous data reads</li> <li>• pool_async_index_reads - Buffer pool asynchronous index reads</li> <li>• pool_async_xda_reads - Buffer pool asynchronous XDA data reads</li> <li>• pool_async_col_reads - Buffer pool asynchronous column-organized reads</li> </ul> <p>If the sum of asynchronous reads is not greater than zero, NULL is returned.</p>

Table 80. Information returned by the MON\_BP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
TOTAL_WRITES	BIGINT	<p>The number of times a data, index, or data page for an XML storage object (XDA) was physically written to disk.</p> <p>This is calculated as <math>(pool\_data\_writes + pool\_index\_writes + pool\_xda\_writes + pool\_col\_writes)</math> where <math>pool\_data\_writes</math>, <math>pool\_index\_writes</math>, <math>pool\_xda\_writes</math>, and <math>pool\_col\_writes</math> represent the following monitor elements:</p> <ul style="list-style-type: none"> <li>• <math>pool\_data\_writes</math> - Buffer pool data writes</li> <li>• <math>pool\_index\_writes</math> - Buffer pool index writes</li> <li>• <math>pool\_xda\_writes</math> - Buffer pool XDA data writes</li> <li>• <math>pool\_col\_writes</math> - Buffer pool column-organized writes</li> </ul>
AVG_WRITE_TIME	BIGINT	<p>Average time, in milliseconds, spent physically writing pages from the buffer pool to disk.</p> <p>If the sum of write operations is greater than zero, this is calculated as <math>pool\_write\_time / (pool\_data\_writes + pool\_index\_writes + pool\_xda\_writes + pool\_col\_writes)</math> where <math>pool\_write\_time</math>, <math>pool\_data\_writes</math>, <math>pool\_index\_writes</math>, <math>pool\_xda\_writes</math>, and <math>pool\_col\_writes</math> represent the following monitor elements:</p> <ul style="list-style-type: none"> <li>• <math>pool\_write\_time</math> - Total buffer pool physical write time</li> <li>• <math>pool\_data\_writes</math> - Buffer pool data writes</li> <li>• <math>pool\_index\_writes</math> - Buffer pool index writes</li> <li>• <math>pool\_xda\_writes</math> - Buffer pool XDA data writes</li> <li>• <math>pool\_col\_writes</math> - Buffer pool column-organized writes</li> </ul> <p>If the sum of write operations is not greater than zero, NULL is returned.</p>
SYNC_WRITES_PERCENT	DECIMAL(5,2)	Percentage of write operations that are synchronous.

Table 80. Information returned by the MON\_BP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
GBP_DATA_HIT_RATIO_PERCENT	DECIMAL(5,2)	Group bufferpool data hit ratio. The percentage of time that the database manager did not need to load a page from disk in order to service a data page request because the page was already in the group bufferpool. Outside of a Db2 pureScale environment, this value is null.
GBP_INDEX_HIT_RATIO_PERCENT	DECIMAL(5,2)	Group bufferpool index hit ratio. The percentage of time that the database manager did not need to load a page from disk in order to service an index page request because the page was already in the group bufferpool. Outside of a Db2 pureScale environment, this value is null.
GBP_XDA_HIT_RATIO_PERCENT	DECIMAL(5,2)	Group bufferpool auxiliary storage object hit ratio. The percentage of time that the database manager did not need to load a page from disk in order to service a data page request for XML storage object (XDAs) because the page was already in the group bufferpool. Outside of a Db2 pureScale environment, this value is null.
GBP_COL_HIT_RATIO_PERCENT	DECIMAL(5,2)	Group bufferpool column-organized table data hit ratio. The percentage of time that the database manager did not need to load a page from disk in order to service a column-organized table data request because the page was already in the group bufferpool. Outside of a Db2 pureScale environment, this value is null.
CACHING_TIER_DATA_HIT_RATIO_PERCENT	DECIMAL(5,2)	Reserved for future use.
CACHING_TIER_INDEX_HIT_RATIO_PERCENT	DECIMAL(5,2)	Reserved for future use.
CACHING_TIER_XDA_HIT_RATIO_PERCENT	DECIMAL(5,2)	Reserved for future use.
CACHING_TIER_COL_HIT_RATIO_PERCENT	DECIMAL(5,2)	Reserved for future use.
AVG_SYNC_READ_TIME	BIGINT	Average time, in milliseconds, spent in synchronous reading from bufferpool.
AVG_ASYNC_READ_TIME	BIGINT	Average time, in milliseconds, spent in asynchronous reading from bufferpool.

Table 80. Information returned by the MON\_BP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
AVG_SYNC_WRITE_TIME	BIGINT	Average time, in milliseconds, spent in synchronous writing to the bufferpool.
AVG_ASYNC_WRITE_TIME	BIGINT	Average time, in milliseconds, spent asynchronous writing to the bufferpool.

## MON\_CONNECTION\_SUMMARY - Retrieve metrics for all connections

The MON\_CONNECTION\_SUMMARY administrative view returns key metrics for all connections in the currently connected database.

This view is designed to help monitor the system in a high-level manner, showing incoming work per connection.

The metrics returned represent the accumulation of all metrics for requests that were submitted by the identified connection across all members of the database.

The schema is SYSIBMADM.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the MON\_CONNECTION\_SUMMARY administrative view
- CONTROL privilege on the MON\_CONNECTION\_SUMMARY administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Information returned

Table 81. Information returned by the MON\_CONNECTION\_SUMMARY administrative view

Column name	Data type	Description or Monitor element
APPLICATION_HANDLE	BIGINT	application_handle - Application handle
APPLICATION_NAME	VARCHAR(128)	appl_name - Application name
APPLICATION_ID	VARCHAR(128)	appl_id - Application ID
SESSION_AUTH_ID	VARCHAR(128)	session_auth_id - Session authorization ID
TOTAL_APP_COMMITS	BIGINT	total_app_commits - Total application commits monitor elements
TOTAL_APP_ROLLBACKS	BIGINT	total_app_rollback - Total application rollbacks monitor element

Table 81. Information returned by the MON\_CONNECTION\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
ACT_COMPLETED_TOTAL	BIGINT	act_completed_total - Total completed activities monitor element
APP_RQSTS_COMPLETED_TOTAL	BIGINT	app_rqsts_completed_total - Total application requests completed monitor element
AVG_RQST_CPU_TIME	BIGINT	Average amount of CPU time, in microseconds, used by all external requests that completed successfully. It represents the total of both user and system CPU time. Formula to calculate ratio: TOTAL_CPU_TIME / APP_RQSTS_COMPLETED_TOTAL
ROUTINE_TIME_RQST_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent working on requests that was spent executing user routines. Formula to calculate ratio: TOTAL_ROUTINE_TIME / TOTAL_RQST_TIME
RQST_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of the time spent working on requests that was spent waiting within the database server. Formula to calculate ratio: TOTAL_WAIT_TIME / TOTAL_RQST_TIME
ACT_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of the time spent executing activities that was spent waiting within the database server. Formula to calculate ratio: TOTAL_ACT_WAIT_TIME / TOTAL_ACT_TIME
IO_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of the time spent waiting within the database server that was due to I/O operations. This includes time spent performing direct reads or direct writes, and time spent reading data and index pages from the table space to the bufferpool or writing them back to disk. Formula to calculate ratio: (POOL_READ_TIME + POOL_WRITE_TIME + DIRECT_READ_TIME + DIRECT_WRITE_TIME) / TOTAL_WAIT_TIME
LOCK_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of time spent waiting within the database server that was spent waiting on locks. Formula to calculate ratio: LOCK_WAIT_TIME / TOTAL_WAIT_TIME
AGENT_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of time spent waiting within the database server that was spent by an application queued to wait for an agent under concentrator configurations. Formula to calculate ratio: AGENT_WAIT_TIME / TOTAL_WAIT_TIME

Table 81. Information returned by the MON\_CONNECTION\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
NETWORK_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of time spent waiting within the database server that was spent on client-server communications. This includes time spent sending and receiving data over TCP/IP or using the IPC protocol. Formula to calculate ratio: $\frac{(\text{TCPIP\_SEND\_WAIT\_TIME} + \text{TCPIP\_RCV\_WAIT\_TIME} + \text{IPC\_SEND\_WAIT\_TIME} + \text{IPC\_RCV\_WAIT\_TIME})}{\text{TOTAL\_WAIT\_TIME}}$
SECTION_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent executing sections. This includes the time spent performing sorts. Formula to calculate ratio: $\text{TOTAL\_SECTION\_PROC\_TIME} / (\text{TOTAL\_RQST\_TIME} - \text{TOTAL\_WAIT\_TIME})$
SECTION_SORT_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent performing sorts while executing sections. Formula to calculate ratio: $\text{TOTAL\_SECTION\_SORT\_PROC\_TIME} / (\text{TOTAL\_RQST\_TIME} - \text{TOTAL\_WAIT\_TIME})$
COMPILE_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent compiling an SQL statement. This includes explicit and implicit compile times. Formula to calculate ratio: $\frac{(\text{TOTAL\_COMPILE\_PROC\_TIME} + \text{TOTAL\_IMPLICIT\_COMPILE\_PROC\_TIME})}{(\text{TOTAL\_RQST\_TIME} - \text{TOTAL\_WAIT\_TIME})}$
TRANSACT_END_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent performing commit processing or rolling back transactions. Formula to calculate ratio: $\frac{(\text{TOTAL\_COMMIT\_PROC\_TIME} + \text{TOTAL\_ROLLBACK\_PROC\_TIME})}{(\text{TOTAL\_RQST\_TIME} - \text{TOTAL\_WAIT\_TIME})}$

Table 81. Information returned by the MON\_CONNECTION\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
UTILS_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent running utilities. This includes performing <b>runstats</b> , reorganization, and load operations. Formula to calculate ratio: $(TOTAL\_RUNSTATS\_PROC\_TIME + TOTAL\_REORG\_PROC\_TIME + TOTAL\_LOAD\_PROC\_TIME) / (TOTAL\_RQST\_TIME - TOTAL\_WAIT\_TIME)$
AVG_LOCK_WAITS_PER_ACT	BIGINT	The average number of times that applications or connections waited for locks per coordinator activities (successful and aborted). Formula to calculate ratio: $LOCK\_WAITS / (ACT\_COMPLETED\_TOTAL + ACT\_ABORTED\_TOTAL)$
AVG_LOCK_TIMEOUTS_PER_ACT	BIGINT	The average number of times that a request to lock an object timed out per coordinator activities (successful and aborted). Formula to calculate ratio: $LOCK\_TIMEOUTS / (ACT\_COMPLETED\_TOTAL + ACT\_ABORTED\_TOTAL)$
AVG_DEADLOCKS_PER_ACT	BIGINT	The average number of deadlocks per coordinator activities (successful and aborted). Formula to calculate ratio: $DEADLOCKS / (ACT\_COMPLETED\_TOTAL + ACT\_ABORTED\_TOTAL)$
AVG_LOCK_ESCALS_PER_ACT	BIGINT	The average number of times that locks have been escalated from several row locks to a table lock per coordinator activities (successful and aborted). Formula to calculate ratio: $LOCK\_ESCALS / (ACT\_COMPLETED\_TOTAL + ACT\_ABORTED\_TOTAL)$
ROWS_READ_PER_ROWS_RETURNED	BIGINT	The average number of rows read from the table per rows returned to the application. Formula to calculate ratio: $ROWS\_READ / ROWS\_RETURNED$

Table 81. Information returned by the MON\_CONNECTION\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
TOTAL_BP_HIT_RATIO_PERCENT	DECIMAL(5,2)	The percentage of time that the database manager did not need to load a page from disk to service a data request, index page request, or a request to process column-organized table data, including requests for XML storage objects (XDAs). In a Db2 pureScale environment, this value represents the total hit ratio for the local bufferpool. Formula to calculate ratio: (POOL_DATA_LBP_PAGES_FOUND + POOL_INDEX_LBP_PAGES_FOUND + POOL_XDA_LBP_PAGES_FOUND + POOL_COL_LBP_PAGES_FOUND) / (POOL_DATA_L_READS + POOL_TEMP_DATA_L_READS + POOL_INDEX_L_READS + POOL_TEMP_INDEX_L_READS + POOL_XDA_L_READS + POOL_TEMP_XDA_L_READS + POOL_COL_L_READS + POOL_TEMP_COL_L_READS)
TOTAL_GBP_HIT_RATIO_PERCENT	DECIMAL(5,2)	On a Db2 pureScale system, the percentage of time that the database manager did not need to load a page from disk into the local bufferpool to service a data request, index request, XML storage object (XDA) page request, or a request to process column-organized table data, as the page was located in the group bufferpool. This value will always be 0 for systems outside of Db2 pureScale.
TOTAL_CACHING_TIER_HIT_RATIO_PERCENT	DECIMAL(5,2)	Reserved for future use.
CF_WAIT_TIME_PERCENT	DECIMAL(5,2)	On a Db2 pureScale system, the percentage of the total wait time spent waiting for caching facility communications. This value will always be 0 for systems outside of Db2 pureScale.
RECLAIM_WAIT_TIME_PERCENT	DECIMAL(5,2)	On a Db2 pureScale system, the percentage of the total wait time spent waiting for page reclaims. This value will always be 0 for systems outside of Db2 pureScale.
SPACEMAPPAGE_RECLAIM_WAIT_TIME_PERCENT	DECIMAL(5,2)	On a Db2 pureScale system, the percentage of the total wait time spent waiting for space map page reclaims. This value will always be 0 for systems outside of Db2 pureScale.

## MON\_CURRENT\_SQL - Retrieve key metrics for all activities on all members

The MON\_CURRENT\_SQL administrative view returns key metrics for all activities that were submitted on all members of the database and have not yet been completed, including a point-in-time view of currently executing SQL statements (both static and dynamic) in the currently connected database.

You can use the MON\_CURRENT\_SQL administrative view to identify long running activities and prevent performance problems.

This view returns metrics that are aggregated across all members.

The schema is SYSIBMADM.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the MON\_CURRENT\_SQL administrative view
- CONTROL privilege on the MON\_CURRENT\_SQL administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Information returned

Table 82. Information returned by the MON\_CURRENT\_SQL administrative view

Column name	Data type	Description or Monitor element
COORD_MEMBER	SMALLINT	coord_member - Coordinating member
APPLICATION_HANDLE	BIGINT	application_handle - Application handle
APPLICATION_NAME	VARCHAR(128)	appl_name - Application name
SESSION_AUTH_ID	VARCHAR(128)	session_auth_id - Session authorization ID
CLIENT_APPLNAME	VARCHAR(255)	CURRENT CLIENT_APPLNAME special register
ELAPSED_TIME_SEC	INTEGER	The time elapsed since this activity began, in seconds. The value of this column is null when an activity has entered the system but is in a queue and has not started running.
ACTIVITY_STATE	VARCHAR(32)	activity_state - Activity state
ACTIVITY_TYPE	VARCHAR(32)	activity_type - Activity type

Table 82. Information returned by the MON\_CURRENT\_SQL administrative view (continued)

Column name	Data type	Description or Monitor element
TOTAL_CPU_TIME	BIGINT	total_cpu_time - Total CPU time
ROWS_READ	BIGINT	rows_read - Rows read
ROWS_RETURNED	BIGINT	rows_returned - Rows returned
QUERY_COST_ESTIMATE	BIGINT	query_cost_estimate - Query cost estimate
DIRECT_READS	BIGINT	direct_reads - Direct reads from database
DIRECT_WRITES	BIGINT	direct_writes - Direct writes to database
APPLICATION_ID	VARCHAR(128)	appl_id - Application ID
UOW_ID	INTEGER	uow_id - Unit of work ID
ACTIVITY_ID	INTEGER	activity_id - Activity ID
STMT_TEXT	CLOB(2MB)	stmt_text - SQL statement text

### MON\_CURRENT\_UOW - Retrieve metrics for all units of work

The MON\_CURRENT\_UOW administrative view returns key metrics for all units of work that were submitted on all members of the database.

This view identifies long running units of work and can therefore be used to prevent performance problems.

The MON\_CURRENT\_UOW view represents the coordinator perspective, and not individual members.

The schema is SYSIBMADM.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the MON\_CURRENT\_UOW administrative view
- CONTROL privilege on the MON\_CURRENT\_UOW administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

## Information returned

Table 83. Information returned by the MON\_CURRENT\_UOW administrative view

Column name	Data type	Description or Monitor element
COORD_MEMBER	SMALLINT	coord_member - Coordinator member
UOW_ID	INTEGER	uow_id - Unit of work ID
APPLICATION_HANDLE	BIGINT	application_handle - Application handle
APPLICATION_NAME	VARCHAR(128)	appl_name - Application name
SESSION_AUTH_ID	VARCHAR(128)	session_auth_id - Session authorization ID
CLIENT_APPLNAME	VARCHAR(255)	CURRENT CLIENT_APPLNAME special register
ELAPSED_TIME_SEC	INTEGER	The time elapsed since this unit of work began, in seconds.
WORKLOAD_OCCURRENCE_STATE	VARCHAR(32)	workload_occurrence_state - Workload occurrence state
TOTAL_CPU_TIME	BIGINT	total_cpu_time - Total CPU time
TOTAL_ROWS_MODIFIED	BIGINT	The total number of rows inserted, updated or deleted.
TOTAL_ROWS_READ	BIGINT	The total number of rows read from tables.
TOTAL_ROWS_RETURNED	BIGINT	The total number of rows that have been selected and returned to the application.

## Example

The following example retrieves the application handle, the unit of work ID, the elapsed time, and the total number of rows read and rows returned, for all units of work that have been executed for more than 10 seconds.

```
SELECT APPLICATION_HANDLE AS APPL_HANDLE,
 UOW_ID, ELAPSED_TIME_SEC,
 TOTAL_ROWS_READ AS TOTAL_READ,
 TOTAL_ROWS_MODIFIED AS TOTAL_MODIFIED
 FROM SYSIBMADM.MON_CURRENT_UOW
 WHERE ELAPSED_TIME_SEC > 10
 ORDER BY ELAPSED_TIME_SEC DESC
```

The following is an example of output for this query.

```
APPL_HANDLE UOW_ID ELAPSED_TIME_SEC TOTAL_READ TOTAL_MODIFIED

 254 1 750 87460 0
 61 1 194 108 0
 145 4 82 0 34
```

3 record(s) selected.

## MON\_DB\_SUMMARY - Retrieve accumulated metrics across all members of the database

The MON\_DB\_SUMMARY administrative view returns key metrics in the currently connected database.

This view is designed to help monitor the system in a high-level manner by providing a concise summary of the database.

The metrics returned represent the accumulation of metrics across all members of the database.

The schema is SYSIBMADM.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the MON\_DB\_SUMMARY administrative view
- CONTROL privilege on the MON\_DB\_SUMMARY administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Information returned

Table 84. Information returned by the MON\_DB\_SUMMARY administrative view

Column name	Data type	Description or Monitor element
TOTAL_APP_COMMITS	BIGINT	total_app_commits - Total application commits monitor elements
TOTAL_APP_ROLLBACKS	BIGINT	total_app_rollbacks - Total application rollbacks monitor element
ACT_COMPLETED_TOTAL	BIGINT	act_completed_total - Total completed activities monitor element
APP_RQSTS_COMPLETED_TOTAL	BIGINT	app_rqsts_completed_total - Total application requests completed monitor element
AVG_RQST_CPU_TIME	BIGINT	Average amount of CPU time, in microseconds, used by all external requests that completed successfully. It represents the total of both user and system CPU time.
ROUTINE_TIME_RQST_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent working on requests that was spent executing user routines.

Table 84. Information returned by the MON\_DB\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
RQST_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of the time spent working on requests that was spent waiting within the database server.
ACT_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of the time spent executing activities that was spent waiting within the database server.
IO_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of the time spent waiting within the database server that was due to I/O operations. This includes time spent performing direct reads or direct writes, and time spent reading data and index pages from the table space to the bufferpool or writing them back to disk.
LOCK_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of time spent waiting within the database server that was spent waiting on locks.
AGENT_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of time spent waiting within the database server that was spent by an application queued to wait for an agent under concentrator configurations.
NETWORK_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of time spent waiting within the database server that was spent on client-server communications. This includes time spent sending and receiving data over TCP/IP or using the IPC protocol.
SECTION_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent executing sections. This includes the time spent performing sorts.
SECTION_SORT_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent performing sorts while executing sections.
COMPILE_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent compiling an SQL statement. This includes explicit and implicit compile times.

Table 84. Information returned by the MON\_DB\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
TRANSACTION_END_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent performing commit processing or rolling back transactions.
UTILS_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent running utilities. This includes performing <b>runstats</b> , reorganization, and load operations.
AVG_LOCK_WAITS_PER_ACT	BIGINT	The average number of times that applications or connections waited for locks per coordinator activities (successful and aborted).
AVG_LOCK_TIMEOUTS_PER_ACT	BIGINT	The average number of times that a request to lock an object timed out per coordinator activities (successful and aborted).
AVG_DEADLOCKS_PER_ACT	BIGINT	The average number of deadlocks per coordinator activities (successful and aborted).
AVG_LOCK_ESCAPES_PER_ACT	BIGINT	The average number of times that locks have been escalated from several row locks to a table lock per coordinator activities (successful and aborted).
ROWS_READ_PER_ROWS_RETURNED	BIGINT	The average number of rows read from the table per rows returned to the application.
TOTAL_BP_HIT_RATIO_PERCENT	DECIMAL(5,2)	The percentage of time that the database manager did not need to load a page from disk to service a data request, index page request, or a request to process column-organized table data, including requests for XML storage objects (XDAs). In a Db2 pureScale environment, this value represents the total hit ratio for the local bufferpool.

Table 84. Information returned by the MON\_DB\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
TOTAL_GBP_HIT_RATIO_PERCENT	DECIMAL(5,2)	In a Db2 pureScale environment, the percentage of time that the database manager did not need to load a page from disk into the local bufferpool to service a data request, index request, XML storage object (XDA) page request, or a request to process column-organized table data, as the page was located in the group bufferpool. Outside of a Db2 pureScale environment, this value will always be null.
TOTAL_CACHING_TIER_HIT_RATIO_PERCENT	DECIMAL(5,2)	Reserved for future use.
CF_WAIT_TIME_PERCENT	DECIMAL(5,2)	In a Db2 pureScale environment, the percentage of the total wait time spent waiting for caching facility communications. Outside of a Db2 pureScale environment, this value will always be null.
RECLAIM_WAIT_TIME_PERCENT	DECIMAL(5,2)	In a Db2 pureScale environment, the percentage of the total wait time spent waiting for page reclaims. Outside of a Db2 pureScale environment, this value will always be null.
SPACEMAPPAGE_RECLAIM_WAIT_TIME_PERCENT	DECIMAL(5,2)	In a Db2 pureScale environment, the percentage of the total wait time spent waiting for space map page reclaims. Outside of a Db2 pureScale environment, this value will always be null.

## MON\_LOCKWAITS administrative view - Retrieve metrics for applications that are waiting to obtain locks

The MON\_LOCKWAITS administrative view returns information about agents working on behalf of applications that are waiting to obtain locks in the currently connected database.

This administrative view is useful for identifying locking problems.

This administrative view replaces the SNAPLOCKWAIT administrative view, which is deprecated. Version 9.7 Fix Pack 1 and might be discontinued in a future release.

The schema is SYSIBMADM.

## Authorization

One of the following authorizations is required:

- SELECT privilege on the MON\_LOCKWAITS administrative view
- CONTROL privilege on the MON\_LOCKWAITS administrative view

- DATAACCESS authority

### Default PUBLIC privilege

None

### Information returned

Table 85. Information returned by the MON\_LOCKWAITS administrative view

Column name	Data type	Description or Monitor element
LOCK_NAME	VARCHAR(32)	lock_name - Lock nameYou can use the MON_FORMAT_LOCK_NAME routine to format this internal binary lock name and obtain more details regarding the lock, such as the table and table space that a table lock references.
LOCK_OBJECT_TYPE	VARCHAR(32)	lock_object_type - Lock object type waited on
LOCK_WAIT_ELAPSED_TIME	INTEGER	The time elapsed since the agent started waiting to obtain the lock. This value is given in seconds.
TABSCHEMA	VARCHAR(128)	table_schema - Table schema name  For locks that do not reference a table, NULL is returned.
TABNAME	VARCHAR(128)	table_name - Table name  For locks that do not reference a table, NULL is returned.
DATA_PARTITION_ID	INTEGER	data_partition_id - Data Partition identifier  This element is only applicable to partitioned tables and partitioned indexes. When returning lock level information, a value of -1 represents a lock which controls access to the whole table.
LOCK_MODE	VARCHAR(10)	lock_mode - Lock mode
LOCK_CURRENT_MODE	VARCHAR(10)	lock_current_mode - Original lock mode before conversion  If the LOCK_STATUS is not "C" (converting), then a value of NULL is returned.
LOCK_MODE_REQUESTED	VARCHAR(10)	lock_mode_requested - Lock mode requested
REQ_APPLICATION_HANDLE	BIGINT	req_application_handle - Requesting application handle
REQ_AGENT_TID	BIGINT	req_agent_tid - Requesting agent TID
REQ_MEMBER	SMALLINT	req_member - Requesting member
REQ_APPLICATION_NAME	VARCHAR(128)	The name of the application running at the client that is waiting to acquire this lock.
REQ_USERID	VARCHAR(128)	The current authorization ID for the session being used by the application that is waiting to acquire this lock.

Table 85. Information returned by the MON\_LOCKWAITS administrative view (continued)

Column name	Data type	Description or Monitor element
REQ_STMT_TEXT	CLOB(2MB)	SQL statement section that the application waiting to acquire the lock is executing.  For non-SQL activities, a 0-length string value is returned.
HLD_APPLICATION_HANDLE	BIGINT	hld_application_handle - Holding application handle  If the application holding this lock is unknown or cannot be found then a value of NULL is returned.
HLD_MEMBER	SMALLINT	hld_member - Holding member
HLD_APPLICATION_NAME	VARCHAR(128)	The name of the application running at the client that is holding this lock.  If the application holding this lock is unknown or cannot be found then a 0-length string value is returned.
HLD_USERID	VARCHAR(128)	The current authorization ID for the session being used by the application that is holding this lock.
HLD_CURRENT_STMT_TEXT	CLOB(2MB)	SQL statement text that is currently associated with the application that is holding the lock. Note that this is not necessarily the statement that is causing the lock.

## MON\_PKG\_CACHE\_SUMMARY - Retrieve a high-level summary of the database package cache

The MON\_PKG\_CACHE\_SUMMARY administrative view returns key metrics for both static and dynamic SQL statements in the cache, providing a high-level summary of the database package cache.

The metrics returned are aggregated over all executions of the statement across all members of the database.

The schema is SYSIBMADM.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the MON\_PKG\_CACHE\_SUMMARY administrative view
- CONTROL privilege on the MON\_PKG\_CACHE\_SUMMARY administrative view
- DATAACCESS authority

### Default PUBLIC privilege

None

### Information returned

Table 86. Information returned by the MON\_PKG\_CACHE\_SUMMARY administrative view

Column name	Data type	Description or Monitor element
SECTION_TYPE	CHAR(1)	section_type - Section type indicator

Table 86. Information returned by the MON\_PKG\_CACHE\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
EXECUTABLE_ID	VARCHAR(32) FOR BIT DATA	executable_id - Executable ID
NUM_COORD_EXEC	BIGINT	num_coord_exec - Number of executions by coordinator agent
NUM_COORD_EXEC_WITH_METRICS	BIGINT	num_coord_exec_with_metrics - Number of executions by coordinator agent with metrics
TOTAL_STMT_EXEC_TIME	BIGINT	The total amount of time, in milliseconds, spent executing the statement, including nested activities, over all executions of the statement where the metrics have been collected.
AVG_STMT_EXEC_TIME	BIGINT	The average amount of time, in milliseconds, spent executing the statement, including nested activities, over all executions of the statement where the metrics have been collected.
TOTAL_CPU_TIME	BIGINT	The total amount of CPU time, in microseconds, used while within the database manager. This value represents the combined total of both user and system CPU time. It is calculated as the sum of all total_cpu_time - Total CPU time values for the statement.
AVG_CPU_TIME	BIGINT	The average amount of CPU time, in microseconds, spent within the database manager over all executions of the statement where the metrics have been collected.
TOTAL_LOCK_WAIT_TIME	BIGINT	The total elapsed time, in milliseconds, spent waiting for locks. This value is calculated as the sum of all lock_wait_time - Time waited on locks values for the statement.
AVG_LOCK_WAIT_TIME	BIGINT	The average elapsed time, in milliseconds, spent waiting for locks over all executions of the statement where the metrics have been collected.
TOTAL_IO_WAIT_TIME	BIGINT	The total elapsed time, in milliseconds, spent on I/O operations. This value is calculated as the sum of the elapsed time required to perform direct reads or direct writes, plus the elapsed time spent physically reading or writing data and index pages from or to the table space containers.

Table 86. Information returned by the MON\_PKG\_CACHE\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
AVG_IO_WAIT_TIME	BIGINT	The average elapsed time, in milliseconds, spent on I/O operations over all executions of the statement where the metrics have been collected.
PREP_TIME	BIGINT	prep_time - Preparation time
ROWS_READ_PER_ROWS_RETURNED	BIGINT	The average number of rows read per rows returned over all executions of the statement where the metrics have been collected.
AVG_ACT_WAIT_TIME	BIGINT	Average time spent waiting for database activities per statement execution.
AVG_LOCK_ESCALS	BIGINT	Average number of lock escalations per statement execution.
AVG_RECLAIM_WAIT_TIME	BIGINT	Average time spent waiting for page reclaims per statement execution. Outside of a Db2 pureScale environment, this value will always be null.
AVG_SPACEMAPPAGE_RECLAIM_WAIT_TIME	BIGINT	Average time spent waiting for reclaims of spacemap pages per statement execution. Outside of a Db2 pureScale environment, this value will always be null.
STMT_TEXT	CLOB(2MB)	stmt_text - SQL statement text

## MON\_SERVICE\_SUBCLASS\_SUMMARY - Retrieve metrics for all service subclasses

The MON\_SERVICE\_SUBCLASS\_SUMMARY administrative view returns key metrics for all service subclasses in the currently connected database.

This view is designed to help monitor the system in a high-level manner, showing work executed per service class.

The metrics returned represent the accumulation of all metrics for requests that have executed under the indicated service subclass across all members of the database.

The schema is SYSIBMADM.

### Authorization

One of the following authorizations is required:

- SELECT privilege on the MON\_SERVICE\_SUBCLASS\_SUMMARY administrative view
- CONTROL privilege on the MON\_SERVICE\_SUBCLASS\_SUMMARY administrative view
- DATAACCESS authority

## Default PUBLIC privilege

None

## Information returned

Table 87. Information returned by the MON\_SERVICE\_SUBCLASS\_SUMMARY administrative view

Column name	Data type	Description or Monitor element
SERVICE_SUPERCLASS_NAME	VARCHAR(128)	service_superclass_name - Service superclass name
SERVICE_SUBCLASS_NAME	VARCHAR(128)	service_subclass_name - Service subclass name
SERVICE_CLASS_ID	INTEGER	service_class_id - Service class ID
TOTAL_APP_COMMITS	BIGINT	total_app_commits - Total application commits monitor elements
TOTAL_APP_ROLLBACKS	BIGINT	total_app_rollbacks - Total application rollbacks monitor element
ACT_COMPLETED_TOTAL	BIGINT	act_completed_total - Total completed activities monitor element
APP_RQSTS_COMPLETED_TOTAL	BIGINT	app_rqsts_completed_total - Total application requests completed monitor element
AVG_RQST_CPU_TIME	BIGINT	Average amount of CPU time, in microseconds, used by all external requests that completed successfully. It represents the total of both user and system CPU time.
ROUTINE_TIME_RQST_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent working on requests that was spent executing user routines.
RQST_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of the time spent working on requests that was spent waiting within the database server.
ACT_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of the time spent executing activities that was spent waiting within the database server.
IO_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of the time spent waiting within the database server that was due to I/O operations. This includes time spent performing direct reads or direct writes, and time spent reading data and index pages from the table space to the bufferpool or writing them back to disk.

Table 87. Information returned by the *MON\_SERVICE\_SUBCLASS\_SUMMARY* administrative view (continued)

Column name	Data type	Description or Monitor element
LOCK_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of time spent waiting within the database server that was spent waiting on locks.
AGENT_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of time spent waiting within the database server that was spent by an application queued to wait for an agent under concentrator configurations.
NETWORK_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of time spent waiting within the database server that was spent on client-server communications. This includes time spent sending and receiving data over TCP/IP or using the IPC protocol.
SECTION_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent executing sections. This includes the time spent performing sorts.
SECTION_SORT_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent performing sorts while executing sections.
COMPILE_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent compiling an SQL statement. This includes explicit and implicit compile times.
TRANSACT_END_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent performing commit processing or rolling back transactions.
UTILS_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent running utilities. This includes performing <b>runstats</b> , reorganization, and load operations.
AVG_LOCK_WAITS_PER_ACT	BIGINT	The average number of times that applications or connections waited for locks per coordinator activities (successful and aborted).

Table 87. Information returned by the MON\_SERVICE\_SUBCLASS\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
AVG_LOCK_TIMEOUTS_PER_ACT	BIGINT	The average number of times that a request to lock an object timed out per coordinator activities (successful and aborted).
AVG_DEADLOCKS_PER_ACT	BIGINT	The average number of deadlocks per coordinator activities (successful and aborted).
AVG_LOCK_ESCALS_PER_ACT	BIGINT	The average number of times that locks have been escalated from several row locks to a table lock per coordinator activities (successful and aborted).
ROWS_READ_PER_ROWS_RETURNED	BIGINT	The average number of rows read from the table per rows returned to the application.
TOTAL_BP_HIT_RATIO_PERCENT	DECIMAL(5,2)	The percentage of time that the database manager did not need to load a page from disk to service a data request, index page request, or a request to process column-organized table data, including requests for XML storage objects (XDAs). In a Db2 pureScale environment, this value represents the total hit ratio for the local bufferpool.
TOTAL_GBP_HIT_RATIO_PERCENT	DECIMAL(5,2)	In a Db2 pureScale environment, the percentage of time that the database manager did not need to load a page from disk into the local bufferpool to service a data request, index request, XML storage object (XDA) page request, or a request to process column-organized table data, as the page was located in the group bufferpool. Outside of a Db2 pureScale environment, this value will always be null.
TOTAL_CACHING_TIER_HIT_RATIO_PERCENT	DECIMAL(5,2)	Reserved for future use.
CF_WAIT_TIME_PERCENT	DECIMAL(5,2)	In a Db2 pureScale environment, the percentage of the total wait time spent waiting for caching facility communications. Outside of a Db2 pureScale environment, this value will always be null.
RECLAIM_WAIT_TIME_PERCENT	DECIMAL(5,2)	In a Db2 pureScale environment, the percentage of the total wait time spent waiting for page reclaims. Outside of a Db2 pureScale environment, this value will always be null.

Table 87. Information returned by the MON\_SERVICE\_SUBCLASS\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
SPACEMAPPAGE_RECLAIM_WAIT_TIME_PERCENT	DECIMAL(5,2)	In a Db2 pureScale environment, the percentage of the total wait time spent waiting for space map page reclaims. Outside of a Db2 pureScale environment, this value will always be null.

## MON\_TBSP\_UTILIZATION - Retrieve monitoring metrics for all table spaces and all database partitions

The MON\_TBSP\_UTILIZATION administrative view returns key monitoring metrics, including hit ratios and utilization percentage, for all table spaces and all database partitions in the currently connected database.

### Authorization

This administrative view provides critical information for monitoring performance as well as space utilization.

This administrative view is a replacement for the TBSP\_UTILIZATION administrative view.

The schema is SYSIBMADM.

One of the following authorizations is required:

- SELECT privilege on the MON\_TBSP\_UTILIZATION administrative view
- CONTROL privilege on the MON\_TBSP\_UTILIZATION administrative view
- DATAACCESS authority

### Default PUBLIC privilege

None

### Information returned

Table 88. Information returned by the MON\_TBSP\_UTILIZATION administrative view

Column name	Data type	Description or Monitor element
TBSP_NAME	VARCHAR(128)	tablespace_name - Table space name
MEMBER	SMALLINT	member - Database member
TBSP_TYPE	VARCHAR(10)	tablespace_type - Table space type. This interface returns a text identifier based on defines in sqlutil.h, and is one of: <ul style="list-style-type: none"> <li>• DMS</li> <li>• SMS</li> </ul>

Table 88. Information returned by the MON\_TBSP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
TBSP_CONTENT_TYPE	VARCHAR(10)	tablespace_content_type - Table space content type. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> <li>• ANY</li> <li>• LARGE</li> <li>• SYSTEMP</li> <li>• USRTEMP</li> </ul>
TBSP_STATE	VARCHAR(256)	tablespace_state - Table space state
TBSP_PAGE_SIZE	BIGINT	tablespace_page_size - Table space page size
TBSP_EXTENT_SIZE	BIGINT	tablespace_extent_size - Table space extent size
TBSP_PREFETCH_SIZE	BIGINT	tablespace_prefetch_size - Table space prefetch size
TBSP_USING_AUTO_STORAGE	SMALLINT	tablespace_using_auto_storage - Table space enabled for automatic storage
TBSP_AUTO_RESIZE_ENABLED	SMALLINT	tablespace_auto_resize_enabled - Table space automatic resizing enabled
TBSP_TOTAL_SIZE_KB	BIGINT	The total size of the table space in kilobytes. This is calculated as $(tablespace\_total\_pages * tablespace\_page\_size) / 1024$ where <i>tablespace_total_pages</i> and <i>tablespace_page_size</i> represent the following monitor elements: <ul style="list-style-type: none"> <li>• tablespace_total_pages - Total pages in table space</li> <li>• tablespace_page_size - Table space page size</li> </ul>

Table 88. Information returned by the MON\_TBSP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
TBSP_USABLE_SIZE_KB	BIGINT	<p>The total usable size of the table space, in kilobytes. This equals the total size of the table space minus the space used for overhead pages. This is calculated as <math>(tablespace\_usable\_pages * tablespace\_page\_size) / 1024</math> where <i>tablespace_usable_pages</i> and <i>tablespace_page_size</i> represent the following monitor elements:</p> <ul style="list-style-type: none"> <li>• <i>tablespace_usable_pages</i> - Usable pages in table space</li> <li>• <i>tablespace_page_size</i> - Table space page size</li> </ul>
TBSP_UTILIZATION_PERCENT	DECIMAL(5,2)	<p>The utilization of the table space as a percentage. If <i>tablespace_usable_pages</i> is greater than zero, this is calculated as <math>(tablespace\_used\_pages / tablespace\_usable\_pages) * 100</math> where <i>tablespace_used_pages</i> and <i>tablespace_usable_pages</i> represent the following monitor elements:</p> <ul style="list-style-type: none"> <li>• <i>tablespace_used_pages</i> - Used pages in table space</li> <li>• <i>tablespace_usable_pages</i> - Usable pages in table space</li> </ul> <p>NULL is returned if <i>tablespace_usable_pages</i> is not greater than zero.</p>
TBSP_PAGE_TOP	BIGINT	<i>tablespace_page_top</i> - Table space high watermark
DATA_PHYSICAL_READS	BIGINT	<p>Indicates the number of data pages read from the table space containers (physical) for temporary as well as regular and large table spaces. This is calculated as <math>(pool\_data\_p\_reads + pool\_temp\_data\_p\_reads)</math> where <i>pool_data_p_reads</i> and <i>pool_temp_data_p_reads</i> represent the following monitor elements:</p> <ul style="list-style-type: none"> <li>• <i>pool_data_p_reads</i> - Buffer pool data physical reads</li> <li>• <i>pool_temp_data_p_reads</i> - Buffer pool temporary data physical reads</li> </ul>

Table 88. Information returned by the MON\_TBSP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
DATA_HIT_RATIO_PERCENT	DECIMAL(5,2)	Data hit ratio, that is, the percentage of time that the database manager did not need to load a page from disk to service a data page request.
INDEX_PHYSICAL_READS	BIGINT	<p>Indicates the number of index pages read from the table space containers (physical) for temporary as well as regular and large table spaces. This is calculated as <math>(pool\_index\_p\_reads + pool\_temp\_index\_p\_reads)</math> where <math>pool\_index\_p\_reads</math> and <math>pool\_temp\_index\_p\_reads</math> represent the following monitor elements:</p> <ul style="list-style-type: none"> <li>• <math>pool\_index\_p\_reads</math> - Buffer pool index physical reads</li> <li>• <math>pool\_temp\_index\_p\_reads</math> - Buffer pool temporary index physical reads</li> </ul>
INDEX_HIT_RATIO_PERCENT	DECIMAL(5,2)	Index hit ratio, that is, the percentage of time that the database manager did not need to load a page from disk to service an index data page request.
XDA_PHYSICAL_READS	BIGINT	<p>Indicates the number of data pages for XML storage objects (XDAs) read from the table space containers (physical) for temporary as well as regular and large table spaces. This is calculated as <math>(pool\_xda\_p\_reads + pool\_temp\_xda\_p\_reads)</math> where <math>pool\_xda\_p\_reads</math> and <math>pool\_temp\_xda\_p\_reads</math> represent the following monitor elements:</p> <ul style="list-style-type: none"> <li>• <math>pool\_xda\_p\_reads</math> - Buffer pool XDA data physical reads</li> <li>• <math>pool\_temp\_xda\_p\_reads</math> - Buffer pool temporary XDA data physical reads</li> </ul>

Table 88. Information returned by the MON\_TBSP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
XDA_HIT_RATIO_PERCENT	DECIMAL(5,2)	Auxiliary storage objects hit ratio, that is, the percentage of time that the database manager did not need to load a page from disk to service a data page request for XML storage objects (XDAs). In a Db2 pureScale environment, this value is the percentage of time the database manager used to locate a data page for an XDA in the local buffer pool.
COL_PHYSICAL_READS	BIGINT	Indicates the number of column-organized table data pages read from the physical table space containers for temporary, regular, and large table spaces. This is calculated as $(pool\_col\_p\_reads + pool\_temp\_col\_p\_reads)$ where $pool\_col\_p\_reads$ and $pool\_temp\_col\_p\_reads$ represent the following monitor elements: <ul style="list-style-type: none"> <li>• <math>pool\_col\_p\_reads</math> - Buffer pool column-organized physical reads</li> <li>• <math>pool\_temp\_col\_p\_reads</math> - Buffer pool column-organized temporary physical reads</li> </ul>
COL_HIT_RATIO_PERCENT	DECIMAL(5,2)	Column-organized table data hit ratio. The percentage of time that the database manager did not need to load a page from disk to service a column-organized table data page request. In Db2 pureScale environments, this value is the percentage of time that the database manager located a data page in the local buffer pool.
GBP_DATA_HIT_RATIO_PERCENT	DECIMAL(5,2)	Group bufferpool data hit ratio. The percentage of time that the database manager did not need to load a page from disk in order to service a data page request because the page was already in the group bufferpool. Outside of a Db2 pureScale environment, this value is null.

Table 88. Information returned by the MON\_TBSP\_UTILIZATION administrative view (continued)

Column name	Data type	Description or Monitor element
GBP_INDEX_HIT_RATIO_PERCENT	DECIMAL(5,2)	Group bufferpool index hit ratio. The percentage of time that the database manager did not need to load a page from disk in order to service an index page request because the page was already in the group bufferpool. Outside of a Db2 pureScale environment, this value is null.
GBP_XDA_HIT_RATIO_PERCENT	DECIMAL(5,2)	Group bufferpool auxiliary storage object hit ratio, that is, the percentage of time that the database manager did not need to load a page from disk to service a data page request for XML storage object (XDAs) since the page was in the group bufferpool. Outside of a Db2 pureScale environment, this value will always be null.
GBP_COL_HIT_RATIO_PERCENT	DECIMAL(5,2)	Group bufferpool column-organized table data hit ratio. The percentage of time that the database manager did not need to load a page from disk in order to service a column-organized table data request because the page was already in the group bufferpool. Outside of a Db2 pureScale environment, this value is null.
CACHING_TIER_DATA_HIT_RATIO_PERCENT	DECIMAL(5,2)	Reserved for future use.
CACHING_TIER_INDEX_HIT_RATIO_PERCENT	DECIMAL(5,2)	Reserved for future use.
CACHING_TIER_XDA_HIT_RATIO_PERCENT	DECIMAL(5,2)	Reserved for future use.
CACHING_TIER_COL_HIT_RATIO_PERCENT	DECIMAL(5,2)	Reserved for future use.

### **MON\_TRANSACTION\_LOG\_UTILIZATION administrative view - Retrieve log utilization information**

The MON\_TRANSACTION\_LOG\_UTILIZATION administrative view returns information about transaction log utilization for the currently connected database.

A single row is returned for each member.

## Authorization

One of the following authorizations is required:

- SELECT privilege on the MON\_TRANSACTION\_LOG\_UTILIZATION administrative view
- CONTROL privilege on the MON\_TRANSACTION\_LOG\_UTILIZATION administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

## Information returned

*Table 89. Information returned by the MON\_TRANSACTION\_LOG\_UTILIZATION administrative view*

Column name	Data type	Description or corresponding monitor element
LOG_UTILIZATION_PERCENT	DECIMAL(5,2)	Indicates the percentage of the total log space that is in use.
TOTAL_LOG_USED_KB	BIGINT	total_log_used - Total log space used. This interface returns the value in KB.
TOTAL_LOG_AVAILABLE_KB	BIGINT	total_log_available - Total log available. This interface returns the value in KB.
TOTAL_LOG_USED_TOP_KB	BIGINT	tot_log_used_top - Maximum total log space used. This interface returns the value in KB.
MEMBER	SMALLINT	member - Database member monitor element

## Usage notes

For databases that are configured for infinite logging, the values for LOG\_UTILIZATION\_PERCENT and TOTAL\_LOG\_AVAILABLE\_KB are NULL.

## MON\_WORKLOAD\_SUMMARY - Retrieves metrics for all workloads

The MON\_WORKLOAD\_SUMMARY administrative view returns key metrics for all workloads in the currently connected database.

This view is designed to help monitor the system in a high-level manner, showing incoming work per workload.

The metrics returned represent the accumulation of all metrics for requests that were submitted by connections mapped to the identified workload object across all members of the database.

The schema is SYSIBMADM.

## Authorization

One of the following authorizations is required:

- SELECT privilege on the MON\_WORKLOAD\_SUMMARY administrative view
- CONTROL privilege on the MON\_WORKLOAD\_SUMMARY administrative view
- DATAACCESS authority

## Default PUBLIC privilege

None

## Information returned

Table 90. Information returned by the MON\_WORKLOAD\_SUMMARY administrative view

Column name	Data type	Description or Monitor element
WORKLOAD_NAME	VARCHAR(128)	workload_name - Workload name
WORKLOAD_ID	INTEGER	workload_id - Workload ID
TOTAL_APP_COMMITS	BIGINT	total_app_commits - Total application commits monitor elements
TOTAL_APP_ROLLBACKS	BIGINT	total_app_rollbacks - Total application rollbacks monitor element
ACT_COMPLETED_TOTAL	BIGINT	act_completed_total - Total completed activities monitor element
APP_RQSTS_COMPLETED_TOTAL	BIGINT	Total number of external (application) requests that completed successfully across all members of the database for the specified service subclass
AVG_RQST_CPU_TIME	BIGINT	Average amount of CPU time, in microseconds, used by all external requests that completed successfully. It represents the total of both user and system CPU time.
ROUTINE_TIME_RQST_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent working on requests that was spent executing user routines.
RQST_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of the time spent working on requests that was spent waiting within the database server.

Table 90. Information returned by the MON\_WORKLOAD\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
ACT_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of the time spent executing activities that was spent waiting within the database server.
IO_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of the time spent waiting within the database server that was due to I/O operations. This includes time spent performing direct reads or direct writes, and time spent reading data and index pages from the table space to the bufferpool or writing them back to disk.
LOCK_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of time spent waiting within the database server that was spent waiting on locks.
AGENT_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of time spent waiting within the database server that was spent by an application queued to wait for an agent under concentrator configurations.
NETWORK_WAIT_TIME_PERCENT	DECIMAL(5,2)	The percentage of time spent waiting within the database server that was spent on client-server communications. This includes time spent sending and receiving data over TCP/IP or using the IPC protocol.
SECTION_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent executing sections. This includes the time spent performing sorts.
SECTION_SORT_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent performing sorts while executing sections.
COMPILE_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent compiling an SQL statement. This includes explicit and implicit compile times.

Table 90. Information returned by the MON\_WORKLOAD\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
TRANSACTION_END_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent performing commit processing or rolling back transactions.
UTILS_PROC_TIME_PERCENT	DECIMAL(5,2)	The percentage of time the database server spent actively working on requests that was spent running utilities. This includes performing <b>runstats</b> , reorganization, and load operations.
AVG_LOCK_WAITS_PER_ACT	BIGINT	The average number of times that applications or connections waited for locks per coordinator activities (successful and aborted).
AVG_LOCK_TIMEOUTS_PER_ACT	BIGINT	The average number of times that a request to lock an object timed out per coordinator activities (successful and aborted).
AVG_DEADLOCKS_PER_ACT	BIGINT	The average number of deadlocks per coordinator activities (successful and aborted).
AVG_LOCK_ESCALATIONS_PER_ACT	BIGINT	The average number of times that locks have been escalated from several row locks to a table lock per coordinator activities (successful and aborted).
ROWS_READ_PER_ROWS_RETURNED	BIGINT	The average number of rows read from the table per rows returned to the application.
TOTAL_BP_HIT_RATIO_PERCENT	DECIMAL(5,2)	The percentage of time that the database manager did not need to load a page from disk to service a data request, index page request, or a request to process column-organized table data, including requests for XML storage objects (XDAs). In a Db2 pureScale environment, this value represents the total hit ratio for the local bufferpool.

Table 90. Information returned by the MON\_WORKLOAD\_SUMMARY administrative view (continued)

Column name	Data type	Description or Monitor element
TOTAL_GBP_HIT_RATIO_PERCENT	DECIMAL(5,2)	In a Db2 pureScale environment, the percentage of time that the database manager did not need to load a page from disk into the local bufferpool to service a data request, index request, XML storage object (XDA) page request, or a request to process column-organized table data, as the page was located in the group bufferpool. Outside of a Db2 pureScale environment, this value will always be null.
TOTAL_CACHING_TIER_HIT_RATIO_PERCENT	DECIMAL(5,2)	Reserved for future use.
CF_WAIT_TIME_PERCENT	DECIMAL(5,2)	In a Db2 pureScale environment, the percentage of the total wait time spent waiting for caching facility communications. Outside of a Db2 pureScale environment, this value will always be null.
RECLAIM_WAIT_TIME_PERCENT	DECIMAL(5,2)	In a Db2 pureScale environment, the percentage of the total wait time spent waiting for page reclaims. Outside of a Db2 pureScale environment, this value will always be null.
SPACEMAPPAGE_RECLAIM_WAIT_TIME_PERCENT	DECIMAL(5,2)	In a Db2 pureScale environment, the percentage of the total wait time spent waiting for space map page reclaims. Outside of a Db2 pureScale environment, this value will always be null.

## Monitor procedures and functions

Monitor routines are procedures or functions with names that begin with “MON”, such as MON\_GET\_SERVICE\_SUBCLASS, or “EVMON”, such as EVMON\_FORMAT\_UE\_TO\_TABLES.

### Characteristics of monitor (MON) table functions

These routines perform a variety of different functions related to querying the status of your system, or manipulating monitoring data. Some routines are table functions that return data in the form of a table. For example, numerous table functions provide access to monitor elements. Some monitor routines, notably, the ones that begin with “EVMON” transform data in one format to another. Certain

other routines, such as environment or administration routines, also return information that is relevant when monitoring.

- The "MON" routines are strategically important, therefore the names of these routines will not change in future releases. However, they will have new output columns added when enhancements are made in future releases. Therefore, when you issue a query to retrieve information using a built-in routine, do not use a statement of the form `SELECT * ....` Instead, name the result columns in the `SELECT` statement. This gives the application control over the number of result columns and the sequence in which they are returned.
- Some table functions have two versions, one of which has a `_DETAILS` suffix. The version without the `_DETAILS` suffix provides a relational SQL interface that returns the most commonly used data. The version with the `_DETAILS` suffix provides XML-based access to the monitor data.
- The metrics returned by the monitoring table functions are never reset. They start at 0 when the database is activated and continue to accumulate until the database is deactivated.
- With most table functions, you can choose to receive data for a single object (for example, service class "A") or for all objects.
- As with most table functions, when using these table functions in a Db2 pureScale environment or a partitioned database environment, you can choose to receive data for a single member or for all members. If you choose to receive data for all members, the table functions return one row for each member. You can add the values across members to obtain the value of a monitor element across members.

## Monitor (MON) table functions

Monitor table functions return a diverse set of monitor elements describing system behavior and performance that can be used for understanding, tuning, and troubleshooting applications on the database server. These elements provide information about a diverse set of system performance indicators that can affect application response time. You can also obtain monitor data for a subset of the workload you are interested in.

Some monitor table functions report on various aspects of the overall system workload, for example:

- `MON_GET_CONNECTION` and `MON_GET_CONNECTION_DETAILS`
- `MON_GET_SERVICE_SUBCLASS` and `MON_GET_SERVICE_SUBCLASS_DETAILS`
- `MON_GET_UNIT_OF_WORK` and `MON_GET_UNIT_OF_WORK_DETAILS`
- `MON_GET_WORKLOAD` and `MON_GET_WORKLOAD_DETAILS`
- `MON_GET_DATABASE` and `MON_GET_DATABASE_DETAILS`

Other table functions return data for a specific type of object, for example:

- `MON_GET_APPL_LOCKWAIT`
- `MON_GET_BUFFERPOOL`
- `MON_GET_CONTAINER`
- `MON_GET_EXTENDED_LATCH_WAIT`
- `MON_GET_INDEX`
- `MON_GET_LOCKS`
- `MON_GET_PAGE_ACCESS_INFO`

- MON\_GET\_TABLE
- MON\_GET\_TABLESPACE

Use these table functions to investigate performance issues associated with a particular data object.

Other table functions are useful for subsystem monitoring:

- MON\_GET\_FCM
- MON\_GET\_FCM\_CONNECTION\_LIST
- MON\_GET\_HADR
- MON\_GET\_SERVERLIST
- MON\_GET\_TRANSACTION\_LOG

Other table functions are useful for examining activities and statements:

- MON\_GET\_ROUTINE
- MON\_GET\_AGENT

Other table functions are useful for examining details of individual activities and statements:

- MON\_GET\_ACTIVITY returns details for a specific activity currently running on the system; these details include general activity information (like statement text) and a set of metrics.
- MON\_GET\_INDEX\_USAGE\_LIST returns information from a usage list defined for an index.
- MON\_GET\_TABLE\_USAGE\_LIST returns information from a usage list defined for a table.
- MON\_GET\_PKG\_CACHE\_STMT and MON\_GET\_PKG\_CACHE\_STMT\_DETAILS

In addition, the following table functions serve a progress monitoring role:

- MON\_GET\_AUTO\_MAINT\_QUEUE returns information about all automatic maintenance jobs that are currently queued for execution by the autonomic computing daemon (db2acd).
- MON\_GET\_AUTO\_RUNSTATS\_QUEUE returns information about all objects which are currently queued for evaluation by automatic statistics collection in the currently connected database.
- MON\_GET\_EXTENT\_MOVEMENT\_STATUS returns the status of the extent movement operation.
- MON\_GET\_REBALANCE\_STATUS returns the status of a rebalance operation on a table space.
- MON\_GET\_RTS\_RQST returns information about all real-time statistics requests that are pending in the system, and the set of requests that are currently being processed by the real time statistics daemon.
- MON\_GET\_USAGE\_LIST\_STATUS returns current status on a usage list.

The table functions that begin with MON\_FORMAT\_ return information in an easy-to-read row-based format. The MON\_FORMAT\_LOCK\_NAME takes the internal binary name of a lock and returns detailed information about the lock. The table functions that begin with MON\_FORMAT\_XML\_ take as input an XML metrics document returned by one of the MON\_GET\_\*\_DETAILS table functions (or from the output of statistics, activity, unit of work, or package cache event monitors) and returns formatted row-based output.

- `MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW` returns formatted row-based output on component times.
- `MON_FORMAT_XML_METRICS_BY_ROW` returns formatted row-based output for all metrics.
- `MON_FORMAT_XML_TIMES_BY_ROW` returns formatted row-based output on the combined hierarchy of wait and processing times.
- `MON_FORMAT_XML_WAIT_TIMES_BY_ROW` table function returns formatted row-based output on wait times.

## Event monitor (EVMON) routines

The event monitor routines allow you to access event monitor data, either through an XML document, by using the `EVMON_FORMAT_UE_TO_XML` table function; or through relational tables, by using the `EVMON_FORMAT_UE_TO_TABLES` procedure.

### **EVMON\_FORMAT\_UE\_TO\_TABLES procedure - move an XML document to relational tables**

The `EVMON_FORMAT_UE_TO_TABLES` procedure retrieves data stored in an unformatted event (UE) table produced by an event monitor and converts it into a set of relational tables.

The process of creating relational tables takes place in two steps. First the data in the UE table is converted to XML format, using the `EVMON_FORMAT_UE_TO_XML` table function. This table function is run for you automatically as part of running the `EVMON_FORMAT_UE_TO_TABLES` procedure. Next, the XML document that contains the event monitor data is turned into relational tables using XML decomposition.

## Syntax

```
►►—EVMON_FORMAT_UE_TO_TABLES—(—evmon_type—,—xsrschema—,——————►
►—xsrobjectname—,—xmlschemafilename—,—tabschema—,——————►
►—tblsp_name—,—options—,—commit_count—,—fullselect—)——————►◄
```

The schema is SYSPROC.

## Procedure parameters

### **evmon\_type**

An input parameter of type VARCHAR(128) that represents the type of data stored in the unformatted event table. The possible values are as follows:

#### **LOCKING**

Data stored in the unformatted event table is from a locking event monitor.

#### **PKG\_CACHE**

Data stored in the unformatted event table is from a PACKAGE CACHE event monitor.

**UOW** Data stored in the unformatted event table is from a UOW event monitor.

***xsrschema***

An input parameter of type VARCHAR (128) that specifies the first-part of the name of the XSR object that describes how data from the UE file corresponds to columns in tables. The second-part of the XSR object name is derived from the *xsojectname* parameter. The complete XSR object name is defined as *xsrschema.xsojectname*. If this value is NULL, then the authorization ID of the current session user is used.

***xsojectname***

An input parameter of type VARCHAR (128) that specifies the second-part of the name of the XSR object that describes how data from the UE file corresponds to columns in tables. The first-part of the XSR object name is derived from the *xsrschema* parameter. The complete XSR object name is defined as *xsrschema.xsojectname* and is unique among all objects in the XSR. If this value is NULL, the *xsojectname* is derived as follows: `EVMON_<evmon_type>_SCHEMA_<SQL release level>`.

The XSR object is a copy of the XML schema file that describes the output of the event monitor. It is stored in the XML schema repository (XSR), and defines the relationship between the elements of the interim XML document produced by the first stage of `EVMON_FORMAT_UE_TO_TABLES` processing, and the tables and columns the procedure ultimately produces. The XSR object is also used to manage the mutual dependency between any tables that have been created and the XML schema from which those tables are derived. If the XSR object is dropped, or if any of the tables produced by the procedure are dropped or the columns altered, the dependency between the two is said to be broken. If `EVMON_FORMAT_UE_TO_TABLES` (or the `EVMON_FORMAT_UE_TO_XML` table function) has not yet been run against the UE file for a specific type of event monitor, the XSR object that describes the event monitor output will not yet exist. In this case, the XML schema file for the event monitor is used to create and register an XSR object in the system catalog tables.

If the database has been upgraded to a newer release, the original *xsojectname* must be explicitly specified in order to maintain the dependency between the relational tables and the XML schema.

***xmlschemafile***

An input parameter of type VARCHAR (1024) that is a fully qualified path to the XML schema document on disk that describes the output produced by the event monitor. The XML schema document elements are annotated with information that maps XML elements and attributes to the relational tables and their columns.

This parameter is used register an XSR object. If there is no XSR object registered and enabled for the type of event monitor specified in *evmon\_type*, then an XSR object is registered as follows:

- If *xmlschemafile* is NULL, then the procedure uses the XML schema file on disk that corresponds to value specified for *evmon\_type*, as follows:

***LOCKING***

`sqllib/misc/DB2EvmonLocking.xsd`

***PKG\_CACHE***

`sqllib/misc/DB2EvmonPkgCache.xsd`

***UOW***

`sqllib/misc/DB2EvmonUOW.xsd`

- If you specify the name of an XML schema file, then that file is used to register and enable the XSR object for decomposition.

- If you specify values for the *xsrschema* and *xsrobjectname* parameters, then XSR object is created with these names. Otherwise, the XSR object is named as using the defaults previously described for *xsrobjectname*.

**Important:** If an XSR object has previously been registered and is enabled for decomposition, this parameter is ignored. If you want to register an XSR object using a different XML schema file, you must first drop the existing XSR object.

#### ***tabschema***

An input parameter of type VARCHAR (128) that represents the SQL schema name where the event monitor relational tables are created. If this value is NULL, then the authorization ID of the current session user is used. The SQL schema under which the tables are created is determined as follows:

- If <db2-xdb:SQLSchema> is specified, use this schema;
- If <db2-xdb:defaultSchema> is specified, use this schema;
- If neither of these values is specified, use the value from the *sqlschema* input parameter.

**Note:** When an XML schema is registered for decomposition, the XSR schema repository creates a dependency between each table referenced in the schema and the XSR object that corresponds to this schema. Which means the XSR object name is linked to a unique set of relational tables in the database. If you reference an existing XSR object, its data is always decomposed and inserted into the tables to which the XSR object was linked.

#### ***tbsp\_name***

An input parameter of type VARCHAR(128) that indicates the table space where the relational tables are created. The default value for this parameter is NULL. The table space name specified on the CREATE TABLE statement within the XML schema file takes precedence over this input parameter.

#### ***options***

An input parameter of type VARCHAR(1024) which represents a list of keyword options supported by this table function. Each option must be delimited using a semicolon (;) character. The possible values are:

##### ***RECREATE\_FORCE***

Indicates that the relational tables are dropped and re-created before decomposition.

##### ***RECREATE\_ONERROR***

Indicates that the relational tables are dropped and re-created in the following situations:

1. If the XSR object is not registered, but the tables exist.
2. On the first failed decomposition attempt. Subsequent failures are returned, and no attempts are made to re-create the tables.

If an error occurs, for example, a table space full error or an authorization error, the procedure does not filter the SQLCODE returned by the decomposition procedure. The procedure treats all negative SQLCODES equally and tries to re-create the tables.

##### ***PRUNE\_UE\_TABLE***

Indicates that any binary events that are successfully inserted into relational tables are to be pruned (that is, deleted) from the UE table. Pruning occurs in the same unit of work in which the inserts into the relational tables are performed.

### **UPGRADE\_TABLES**

Indicates that the relational tables produced by this procedure are to be altered so that the table definitions match those defined in the XSR object *xsobjectname* for the current release. Specify this parameter if you want to upgrade any relational tables that were created in an earlier release to reflect any changes made for the current release. The following types of changes might occur from release to release:

- New columns might be added to tables
- New tables might be added to output of the event monitor
- Column definitions might change (For example data type, or length).

If you do not use the UPGRADE\_TABLES option, then the existing table definitions are retained. Data for any new columns or tables added in the current release is not written to the relational tables.

If UPGRADE\_TABLES is specified, the original *xsobjectname* must also be explicitly specified.

### ***commit\_count***

An input parameter of type INTEGER. The possible values are as follows:

- 1** Commit after every 100 successful documents decomposed. -1 is the default value.
- 0** Never commit.
- n*** Commit after every *n* documents successfully decomposed.

### ***fullselect***

An input parameter of type CLOB(2M) that represents the fullselect statement from an unformatted event table. The fullselect statement is a query that conforms to the rules of the SELECT statement. The query must follow the following rules:

- The query must use the "\*" clause or specify all the columns of the unformatted event table. Otherwise an error is returned. The columns must be specified in the same order as returned by the DESCRIBE statement of the unformatted event table.
- The query must select only from an unformatted event table.
- The WHERE clause can use any of the non-LOB columns of the unformatted event table to filter out events.

### **Authorization**

EXECUTE privilege on the EVMON\_FORMAT\_UE\_TO\_TABLES stored procedure.

SELECT privilege on the unformatted event table, if you did not create it.

CREATE privilege to create the relational tables in the specified SQL schema.

INSERT privilege to insert into the relational tables, if you did not create them.

All privileges required by the XDB\_DECOMP\_XMP\_FROM\_QUERY procedure.

### **Default PUBLIC privilege**

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

## Usage notes

### Relationship of records in a UE table to the output of the EVMON\_FORMAT\_UE\_TO\_TABLES table function

There is not a one-to-one mapping between the records written to the UE table and the output of the EVMON\_FORMAT\_UE\_TO\_TABLES procedure. Some events generate multiple records in the UE table; some result in just one record being added. When writing data to relational tables, the EVMON\_FORMAT\_UE\_TO\_TABLES procedure might, in some cases combine information in multiple UE table records into a single relational table, or it may produce more than one row in different output tables.

### Table creation

In order for decomposition to occur, a set of relational tables must exist. The EVMON\_FORMAT\_UE\_TO\_TABLES procedure creates the relational tables automatically, as follows:

- The procedure parses the event monitor XML schema file to find the <db2-mon:createStmt> elements. Each element contains a complete CREATE TABLE statement.
- The procedure extracts and runs the CREATE TABLE statements.

The <db2-mon:createStmt> is a child element of the existing <db2-xdb:table> element. Only the EVMON\_FORMAT\_UE\_TO\_TABLES procedure recognizes and uses this element. All other procedures that parse the XML schema file, such as the XSR objects, ignore this element.

Do not qualify the table name within the <db2-mon:createStmt>.

### XML schema files from release to release

The default XML schema files provided by each event monitor always reflects the XML schema for the current release. So, when you run EVMON\_FORMAT\_UE\_TO\_TABLES (or EVMON\_FORMAT\_UE\_TO\_XML), the output reflects the monitor elements defined for that event monitor in that release. The next section describes what happens if the schema files for the event monitors happen to change over time. Understanding the impact of these changes is important if you create tables using the EVMON\_FORMAT\_UE\_TO\_TABLES procedure, and then apply a fix pack or upgrade to a new release.

### Impact of schema updates on tables produced by EVMON\_FORMAT\_UE\_TO\_TABLES

New monitor elements are likely to be added to event monitors in future fix packs or releases. These new monitor elements might result in new columns or even new tables being produced by the EVMON\_FORMAT\_UE\_TO\_TABLES procedure. However, if you already have tables that were created by this procedure before a fix pack was applied, or before upgrading to a new release, you need to do the following to have the new relational columns or tables created:

#### For fix pack updates

If relational tables produced by EVMON\_FORMAT\_UE\_TO\_TABLES before the installation of the latest fix pack still exist, you must force the creation of a new set of tables based on the new schema shipped in the fix pack if you want to see the new monitor elements in relational format.

To force the EVMON\_FORMAT\_UE\_TO\_TABLES procedure to use the new schema shipped in the fix pack and create new tables, perform the following steps:

1. Break the dependency between the currently registered version of the XML schema (see the note under the *tabschema* parameter of the EVMON\_FORMAT\_UE\_TO\_TABLES procedure for more information about schema registration) and the existing tables by performing one of the following actions:
  - Drop one of the existing tables that were produced by EVMON\_FORMAT\_UE\_TO\_TABLES
  - Drop the registered XML schema object associated with the existing tables using the DROP XSROBJECT statement. For example, to drop the registered XML schema object associated with the tables produced by EVMON\_FORMAT\_UE\_TO\_TABLES for the locking event monitor for Db2 V9.7, use the following command: DROP XSROBJECT EVMON\_LOCKING\_SCHEMA\_SQL09070.
  - Alter any existing column that corresponds to an annotated monitor element in the currently registered XML schema object.
2. Run the EVMON\_FORMAT\_UE\_TO\_TABLES procedure, using the FORCE option. This option causes the old tables to be dropped, and a new set of tables to be produced. If you omit this option, a SQL0601N error is returned.

This process is illustrated in “Example 5: Picking up new elements in a fix pack update” on page 470.

If you do not perform the preceding steps, existing tables are updated based on the previously registered schema file. Any new columns or tables that might have been added in the fix pack are not reflected in the output of the EVMON\_FORMAT\_UE\_TO\_TABLES procedure.

### For release upgrades

Unless you specify otherwise, the default version of the XML schema file for the current release is used when you call the EVMON\_FORMAT\_UE\_TO\_TABLES procedure. So, if you upgrade to a new release of the Db2 product, then, by default, the new version of the schema file is used when you run the procedure.

If tables from the previous release do not exist, EVMON\_FORMAT\_UE\_TO\_TABLES produces tables using the most recent schema. However, if tables from the previous release exist, you must use the FORCE or RECREATE\_ONERROR options to cause the old tables to be replaced by new ones. Otherwise, a SQL0601N error is returned. “Example 6: Picking up new elements in a release update” on page 470 shows an example of recreating the tables using the default schema for a new release.

Alternatively, you can continue to use the existing tables, without adding any new columns or tables that might have been introduced in the latest release. To have the existing tables updated, you must specify the name of the registered XML schema file that was used to create the tables for the *xsrobjectname* parameter of the EVMON\_FORMAT\_UE\_TO\_TABLES procedure.

“Example 7: Using the previous relational tables on a release update” on page 470 shows an example of using the schema from a previous release.

**Note:** You cannot pick up any new columns or tables introduced in fix packs or in new releases while retaining the data that was previously in the relational tables produced by `EVMON_FORMAT_UE_TO_TABLES`. Picking up any new columns requires the tables to be re-created.

### Partial events

If partial or incomplete events exist in the UE table, a message (SQL443N) is returned when you run `EVMON_FORMAT_UE_TO_TABLES`. Incomplete events can occur when an agent finishes processing before the entire event record can be inserted in to the UE table. This situation can sometimes arise where locking is involved, particularly in partitioned database environments. For example, when the LOCKWAIT threshold is exceeded, details about the holder of the lock are written to the UE table. However, details about agents waiting for a lock on the same object are not captured until the lock times out or the waiter acquires the lock. If `EVMON_FORMAT_UE_TO_TABLES` is run before the agent waiting for the lock has written its information, then only a part of the information about the lock might exist in the UE table.

To see details about the incomplete events, run `EVMON_FORMAT_UE_TO_XML` with the `LOG_PARTIAL_EVENTS` option.

### Examples

- “Example 1: Using default parameters”
- “Example 2: Attempting to use tables under a different schema” on page 469
- “Example 3: Attempting to use tables under a different schema” on page 469
- “Example 4: Using the `RECREATE_FORCE` option” on page 469
- “Example 5: Picking up new elements in a fix pack update” on page 470
- “Example 6: Picking up new elements in a release update” on page 470
- “Example 7: Using the previous relational tables on a release update” on page 470
- “Example 8: Using the `UPGRADE_TABLES` and `PRUNE_UE_TABLE` options” on page 470

### Example 1: Using default parameters

A user named Paul calls the procedure using the default parameters and requires all events that are part of the service class `STUDENTS` to be inserted into the relational tables.

```
EVMON_FORMAT_UE_TO_TABLES (
 'UOW', NULL, NULL, NULL, NULL, NULL, NULL, -1,
 'SELECT * FROM UOWUE
 WHERE service_subclass_name = 'STUDENTS'
 ORDER BY event_id, event_timestamp')
```

The results of the call are as follows:

1. The procedure parses the `DB2EvmonUOW.xsd` file, which is the default XML schema file, to identify the set of relational tables to create.
2. The relational tables are created under SQL schema Paul.

3. The XML schema is registered with an XSR object name of PAUL.EVMON\_UOW\_SCHEMA\_SQL09070
4. XSR object is enabled for decomposition.
5. Data is decomposed and inserted into the tables under SQL schema Paul.

### Example 2: Attempting to use tables under a different schema

In a continuation of the previous example, a user named Dave calls the stored procedure, setting the *tabschema* parameter to Paul.

```
EVMON_FORMAT_UE_TO_TABLES (
 'UOW', NULL, NULL, NULL, 'Paul', NULL, NULL, -1,
 'SELECT * FROM UOWTBLE
 ORDER BY event_timestamp')
```

The results of the call are as follows:

1. The procedure parses the DB2EvmonUOW.xsd file, which is the default XML schema file, to identify the set of relational tables to create.
2. The procedure attempts to create the tables under schema Paul. However, an error is returned because the relational tables currently exist under the SQL schema PAUL. Previously existing tables cannot be used when a new XSR object is being registered.

### Example 3: Attempting to use tables under a different schema

In a continuation of the previous example, a user named Greg calls the stored procedure setting the input parameter *xsrschema* to Paul.

```
EVMON_FORMAT_UE_TO_TABLES (
 'UOW', 'Paul', NULL, NULL, NULL, NULL, NULL, -1,
 'SELECT * FROM UOWTBL
 ORDER BY event_timestamp')
```

The results of the call are as follows:

1. The XSR object Paul.EVMON\_UOW\_SCHEMA\_SQL09070, which exists, is enabled for decomposition.
2. If Greg has INSERT privileges on the tables, then data is decomposed and inserted into the relational tables under SQL schema Paul. The existing XSR object Paul.EVMON\_UOW\_SCHEMA\_SQL09070 is used, so the SQL schema for the relational tables is obtained from the XSR object, instead of being provided as an input parameter to the procedure.

### Example 4: Using the RECREATE\_FORCE option

In a continuation of the previous example, Paul wants to re-create the tables again, but in table space MYSPACE. Paul calls the procedure with the RECREATE\_FORCE option and the *tbsp\_name* parameter.

```
EVMON_FORMAT_UE_TO_TABLES (
 'UOW', NULL, NULL, NULL, NULL, 'MYSPACE', 'RECREATE_FORCE', -1,
 'SELECT * FROM UOWTBL
 ORDER BY event_timestamp')
```

The results of the call are as follows:

1. The XSR object Paul.EVMON\_UOW\_SCHEMA\_SQL09070, which exists, is enabled for decomposition.
2. The RECREATE\_FORCE option is set.

3. The XML schema file is retrieved from the schema repository and parsed to identify the set of relational files.
4. The current tables are dropped and created again in the MYSPACE table space.
5. Data is decomposed and inserted into the new tables.

### Example 5: Picking up new elements in a fix pack update

A new XML element called “db2EventNew” has been added to the XML schema file of the locking event monitor in the latest fix pack. Paul wants to pick up the new element to use in the decomposition of an XML file. To do so, he follows the following steps:

1. Paul drops the XSR object created in the original release:
2. He calls the procedure with the RECREATE\_ONERROR option.

```

DROP XSROBJECT EVMON_LOCKING_SCHEMA_SQL09070

EVMON_FORMAT_UE_TO_TABLES (
 'LOCKING', NULL, NULL, NULL, NULL, NULL, 'RECREATE_ONERROR', -1,
 'SELECT * FROM LOCK
 ORDER BY event_timestamp')

```

The results of the call are as follows:

- a. The XSR object does not exist, so the default DB2EvmonLocking.xsd schema file is parsed to identify the set of relational tables.
- b. As the RECREATE\_ONERROR option was specified, the existing tables are dropped and re-created.

### Example 6: Picking up new elements in a release update

Paul is upgrading to a new product release and wants to pick up the new changes in the event monitor XML schema file. Paul calls the procedure with the RECREATE\_ONERROR option.

```

EVMON_FORMAT_UE_TO_TABLES (
 'LOCKING', NULL, NULL, NULL, NULL, NULL, 'RECREATE_ONERROR', -1,
 'SELECT * FROM LOCK
 ORDER BY event_timestamp')

```

The results of the call are as follows:

1. The XSR object Paul.EVMON\_LOCKING\_SCHEMA\_SQL1000 does not exist.
2. As the RECREATE\_ONERROR option was specified, the tables are dropped and re-created.

### Example 7: Using the previous relational tables on a release update

Greg has upgraded to a new product release. Greg calls the procedure with the *xsrobjectname* value from the previous release.

```

EVMON_FORMAT_UE_TO_TABLES (
 'LOCKING', NULL, 'EVMON_LOCKING_SCHEMA_SQL09070', NULL, NULL, NULL, -1,
 'SELECT * FROM LOCK
 ORDER BY event_timestamp')

```

### Example 8: Using the UPGRADE\_TABLES and PRUNE\_UE\_TABLE options

Paul created a unit of work event monitor in V9.7 that writes its output to a UE table called UOWTABLE. He then upgrades to V10.1 and wants the relational tables produced in the previous release by EVMON\_FORMAT\_UE\_TO\_TABLES to be upgraded using the UPGRADE\_TABLES option. This upgrade occurs before the new

data is processed. Furthermore he wants to have the records from UOWTABLE deleted using the PRUNE\_UE\_TABLE option after they have been processed.

```

EVMON_FORMAT_UE_TO_TABLES (
'UOW', NULL, 'EVMON_UOW_SCHEMA_SQL09070', NULL, NULL, NULL,
'UPGRADE_TABLES;PRUNE_UE_TABLE', -1,
'SELECT * FROM UOWTABLE
ORDER BY event_timestamp')

```

**Note:** In this example, the value 'EVMON\_UOW\_SCHEMA\_SQL09070' must be specified for the *xsobjectname* parameter, since 'EVMON\_UOW\_SCHEMA\_SQL09070' is the name of the XSR object that was used in the most recent release where EVMON\_FORMAT\_UE\_TO\_TABLES was run to create relational tables from a UE table.

## Information returned

There is no output from the procedure except the SQLCA. The SQLCA indicates the completion status. The possible SQLCODES are:

- 0** All events were successfully inserted into the relational tables.
- 16278** One or more events were not inserted into the relational tables. The tokens within the SQLCA contain the total number of documents that were attempted and the total number of documents that succeeded decomposition.

A diagnostic file is also created; and the name and location of that diagnostic file is stored in the db2diag log files, located in the diagnostic path.

### *negative sqlcode*

An error has occurred, and investigating the SQLCODE message can provide additional details regarding the failure. For additional diagnostic messages, see the db2diag log files located in the diagnostic path.

## **EVMON\_FORMAT\_UE\_TO\_XML table function - convert unformatted events to XML**

The EVMON\_FORMAT\_UE\_TO\_XML table function extracts binary events from an unformatted event table and formats them into an XML document.

## Syntax

```

►► EVMON_FORMAT_UE_TO_XML (—options—, —————►
► FOR EACH ROW OF (—fullselect-statement—) —————►►

```

The schema is SYSPROC.

## Table function parameters

### **options**

An input argument of type VARCHAR(1024) that represents a list of keyword options supported by this table function.

### **LOG\_TO\_FILE**

Indicates that the table function is to write the XML document to a file if the XML document is greater than 100 MB. The maximum size of each document returned by this table function per row is 100 MB. The file is written to the <xml\_document\_id>.xml file, where

<xml\_document\_id> is the unique ID generated for each document. The output file is written to the diagnostic path directory.

#### **LOG\_PARTIAL\_EVENTS**

Indicates that the table function is to write all partial (incomplete) events to a file. See the “Usage notes” on page 473 section of this topic for more information about partial events.

#### **SUPPRESS\_PARTIAL\_EVENTS\_ERR**

Suppresses the SQL443N warning message when partial or incomplete events exist in the unformatted event (UE) table. For more information about partial events, see the “Usage notes” on page 473 section.

**NULL** No options selected.

#### ***fullselect-statement***

The fullselect statement is a query that conforms to the rules of the SELECT statement. The query must follow the following rules:

- The query must use the "\*" clause or specify all the columns of the unformatted event table. Otherwise an error is returned. The columns must be specified in the same order as returned by the DESCRIBE statement of the unformatted event table.
- The query must select only from an unformatted event table.
- The WHERE clause can use any of the non-LOB columns of the unformatted event table to filter out events.
- The SELECT statement must be specified by the keyword FOR EACH ROWS OF, enclosed in brackets.

### **Authorization**

EXECUTE privilege on the EVMON\_FORMAT\_UE\_TO\_XML function.

SELECT privilege on the unformatted event table.

### **Default PUBLIC privilege**

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

### **Examples**

*Example 1:* Query all events from the unformatted event table “MYLOCKS”.

```
SELECT evmon.* FROM TABLE (
 EVMON_FORMAT_UE_TO_XML (
 NULL,
 FOR EACH ROW OF (
 select * from MYLOCKS
 order by EVENT_TIMESTAMP)))
AS evmon;
```

*Example 2:* Query all events of type “LOCKWAIT” that have occurred in the last 5 hours from the unformatted event table “LOCK”.

```
SELECT evmon.* FROM TABLE (
 EVMON_FORMAT_UE_TO_XML (
 NULL,
 FOR EACH ROW OF (

```

```

select * from LOCK order by EVENT_TIMESTAMP
where EVENT_TYPE = 'LOCKWAIT'
and EVENT_TIMESTAMP >= CURRENT_TIMESTAMP - 5 hours)))
AS evmon;

```

*Example 3:* Get all events that belong to workload “PAYROLL” that occurred in the last 32 hours from the unformatted event table “UOW”. Write the result to a file if any document is greater than 100 MB.

```

SELECT evmon.* FROM TABLE (
 EVMON_FORMAT_UE_TO_XML(
 'LOG TO FILE',
 FOR EACH ROW OF (
 select * from UOW order by EVENT_TIMESTAMP
 where WORKLOAD_NAME = 'PAYROLL'
 and EVENT_TIMESTAMP = CURRENT_TIMESTAMP - 32 hours)))
AS evmon;

```

*Example 4:* Query all unit of work events from the “UOWEVMON” table, and use the XMLTABLE table function to present the UOW ID, UOW start and stop times, and the user ID for the person who issued the unit of work.

```

SELECT EVENT.UOW_ID, EVENT.APPLICATION_ID, EVENT.SESSION_AUTHID,
EVENT.START_TIME, EVENT.STOP_TIME
FROM TABLE(
 EVMON_FORMAT_UE_TO_XML(
 'LOG TO FILE',
 FOR EACH ROW OF(
 select * from UOWEVMON)))
AS UEXML,
XMLTABLE(
 XMLNAMESPACES(DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'),
 '$uowevent/db2_uow_event'
 PASSING XMLPARSE(DOCUMENT UEXML.XMLREPORT) as "uowevent"
 COLUMNS UOW_ID INTEGER PATH 'uow_id',
 MEMBER SMALLINT PATH '@member',
 APPLICATION_ID VARCHAR(128) PATH 'application_id',
 SESSION_AUTHID VARCHAR(128) PATH 'session_authid',
 START_TIME TIMESTAMP PATH 'start_time',
 STOP_TIME TIMESTAMP PATH 'stop_time'
)
AS EVENT

```

## Usage notes

### Impact of the EVMON\_FORMAT\_UE\_TO\_XML table function on memory usage

Depending on the event monitor type that produced the UE table, the EVMON\_FORMAT\_UE\_TO\_XML table function might map multiple records from the unformatted event table into a single event. In such a case, the records are cached in memory until all the records that make up the event are received. A larger memory requirement might result if the records passed into the table function are not in the order they were created and inserted into the table. If the records are not sorted in this manner, the table function must cache records for multiple events. To avoid this issue, qualify the *fullselect-statement* parameter with an ORDER BY clause that contains the following columns: EVENT\_ID, EVENT\_TIMESTAMP, EVENT\_TYPE, and MEMBER. Memory consumption is reduced because at any particular time, the table function is processing and caching records from only a single event.

### Relationship of records in a UE table to the output of the EVMON\_FORMAT\_UE\_TO\_XML table function

There is not a one-to-one mapping between the records written to the UE table and the output of the EVMON\_FORMAT\_UE\_TO\_XML table function. Some events generate multiple records in the UE table; some result in just one record being added. The EVMON\_FORMAT\_UE\_XML table function always combines all records from a UE table that describe a single event into one XML document.

### Partial events

If partial or incomplete events exist in the UE table, a message (SQL443N) is returned when you run EVMON\_FORMAT\_UE\_TO\_XML, whether or not you specify the LOG\_PARTIAL\_EVENTS option. You can suppress the message by using the SUPPRESS\_PARTIAL\_EVENTS\_ERR option. Incomplete events can occur when an agent finishes processing before the entire event record can be inserted in to the UE table. This situation can sometimes arise where locking is involved, particularly in partitioned database environments. For example, when the LOCKWAIT threshold is exceeded, details about the holder of the lock are written to the UE table. However, details about agents waiting for a lock on the same object are not captured until the lock times out or the waiter acquires the lock. If EVMON\_FORMAT\_UE\_TO\_XML is run before the agent waiting for the lock has written its information, then only a part of the information about the lock might exist in the UE table.

When you specify the LOG\_PARTIAL\_EVENTS option, incomplete events in the UE table are written to a *separate* XML document. In addition, a message is written to the db2diag log files indicating that an incomplete event took place. The message specifies the file name of the XML document that contains details about the incomplete event. The XML documents produced can be formatted using the db2evmonfmt tool.

### Information returned

Table 91. Information returned for EVMON\_FORMAT\_UE\_TO\_XML.

Column Name	Data Type	Description or corresponding monitor element
XMLID	VARCHAR(1024)	xmldid - XML ID monitor element
XMLREPORT	BLOB(100M)	An XML document containing a single complete event. Each document has a maximum size of 100 MB.

### EVMON\_UPGRADE\_TABLES procedure - Upgrade event monitor target tables

The EVMON\_UPGRADE\_TABLES procedure alters event monitor target SQL or unformatted event tables to accommodate new or changed monitoring elements that have been added since the event monitor was created.

### Syntax

```

▶▶—EVMON_UPGRADE_TABLES—(—evmon_name—,—evmon_type—▶
▶,—options—,—num_evmons_evaluated—▶
▶,—num_evmons_to_upgrade—,—num_evmons_upgraded—)▶▶

```

The schema is SYSPROC.

## Procedure parameters

### *evmon\_name*

An input argument of type VARCHAR(128) that specifies the name of the event monitor for which existing table definitions are to be upgraded. The event monitor target type must be table or unformatted event table. If *evmon\_name* is specified, any *evmon\_type* argument is ignored.

The name may be a pattern-expression similar to that used in the LIKE predicate which means it can include underscore (\_) or percent (%) characters as wild card characters. For more information about the LIKE predicate, see LIKE predicate. If the pattern-expression needs to include either the underscore or the percent character, the escape option is used to specify a character to precede either the underscore or the percent character in the pattern.

If *evmon\_name* is not provided or set to NULL, all event monitors with table or unformatted event table output targets will be processed, subject to criteria supplied by the other input parameters.

### *evmon\_type*

An input argument of type VARCHAR(128) that specifies the type of event monitor for which existing table definitions are to be upgraded. The type specified must be one of the values in the TYPE column of SYSCAT.EVENTS. Refer to SYSCAT.EVENTS catalog view for details. Any event monitor for this event type and having target type of table or unformatted event table will be evaluated for upgrading. If *evmon\_type* is specified, *evmon\_name* must be NULL. If *evmon\_type* is not provided or set to NULL, all types of event monitors will be evaluated, subject to the criteria supplied by the other input parameters.

### *options*

An input argument of type CLOB(8K) that enables you to specify one or more event monitor upgrade settings. This parameter defaults to NULL. Settings are specified as name value pairs using the format:

<setting name tag>value</setting name tag>

Each setting can be specified a maximum of one time. Setting names are case sensitive. The setting values are case insensitive unless otherwise noted. The available setting name tags are as follows.

- '<tblspaceName>value</tblspaceName>'

Specifies the name of a table space into which any new groups are to be created. This value is case sensitive. If this is not specified and an event monitor's current target tables are all in the same table space then any new table will be created in that same table space (so that all tables are in the same table space). Otherwise the algorithm for picking a table space as described for the "IN" clause for "CREATE EVENT MONITOR" is used (see CREATE EVENT MONITOR statement).

- '<createNewGroups>value</createNewGroups>'

Specifies whether or not new groups are to be created. The value can be either "yes" or "no". If not specified it defaults to "yes" and any new group will be created for an event monitor being upgraded.

- '<verbose>value</verbose>'

Specifies whether or not to return diagnostics from the stored procedure as a result set. The value can be either "yes" or "no". If not specified it defaults to "yes" and diagnostics are return in the result set. If "no" is specified no result

set is passed back. Note that a user temporary table space must exist in order for the stored procedure to return a result set. See Table 1 for result set.

- '<force>value</force>'

Specifies whether or not to force the checking of event monitors to determine if upgrading is required. If not specified it defaults to "no". When "no" is specified the VERSIONNUMBER column of SYSCAT.EVENTMONITORS is checked; if its value equals the current version then the event monitor is considered up to date. If "yes" is specified the VERSIONNUMBER column of SYSCAT.EVENTMONITORS is ignored and the event monitor's tables are rechecked to see if upgrading is required or if new tables are to be added. Setting force to yes is useful when an event monitor which has had a new event group added to it has already been upgraded once using the createNewGroups option with a value of NO and the need is to force that event monitor to be upgraded again so that a new table is created for the new event group.

- '<escape>char</escape>'

Specifies a character to be used to modify the special meaning of the underscore (\_) and percent (%) characters in the pattern-expression of the evmon\_name. This allows the evmon\_name to be used to match values that contain the actual percent and underscore characters. If this option is not specified no escape character is provided.

#### ***num\_evmons\_evaluated***

This output parameter of type INTEGER returns the number of event monitors that were checked by the stored procedure to determine if their tables required upgrading.

#### ***num\_evmons\_to\_upgrade***

This output parameter of type INTEGER returns the total number of event monitors that had at least one target table that actually required upgrading. This includes event monitors having new or modified elements or new event groups.

#### ***num\_evmons\_upgraded***

This output parameter of type INTEGER returns the total number of event monitors whose target tables that were successfully upgraded or created.

### **Authorization**

DBADM authority is required to execute the function.

### **Default PUBLIC privilege**

None

### **Usage notes**

- If evmon\_name and evmon\_type are both NULL, then the stored procedure will scan the SYSCAT.EVENTTABLES catalog table and check each TABLE and UE Table event monitor to see if it's target tables require upgrading.
- The VERSIONNUMBER column of SYSCAT.EVENTMONITORS contains the version, release, and modification level in which the event monitor was created or last upgraded. An event monitor target table is considered to require upgrading if columns were added or modified between the VERSIONNUMBER and the current release. That is, only changes made between these releases will be taken into consideration.

- During database upgrade if the VERSIONNUMBER column is not filled in it will be filled with the release number from which the event monitor is being upgraded.
- If evmon\_upgrade\_tables processes an event monitor and that event monitor requires no changes then VERSIONNUMBER is updated to the current level. This makes it easy to see which event monitors have been processed.
- If evmon\_type is specified then the stored procedure will scan the SYSCAT.EVENTTABLES catalog table and check each event monitor that records that event type to see if its target tables require upgrading. If an event monitor can record more than one event type and one of its types is selected for upgrading then only the tables for that type are checked and upgraded if necessary. For example, if an event monitor is created CREATE EVENT MONITOR SAMPLE\_EVMON FOR CONNECTIONS, STATEMENTS WRITE TO TABLE and CONNECTIONS is specified for the evmon\_type, then only groups CONTROL, CONNHEADER and CONN are checked; STMT and SUBSECTION are skipped but the event monitor version number is updated regardless. If later it was required to upgrade the statement event tables, then the force option would need to be used.
- Before a table is checked for an upgrade, it is first validated with the same rules used when an event monitor activates. This validation ensures that each column name matches a monitor element name, that the data type for each column is compatible with that element, and that, in a partitioned database environment, the first column of the table is PARTITION\_KEY.
- If a new logical data group is to be added to the event monitor, ALTER EVENT MONITOR is called to add the logical data group. The new table's schema then defaults to the value in the CURRENT\_SCHEMA special register.
- For any event monitor with tables that require upgrading:
  - If the event monitor is active it is deactivated (the procedure waits for queued events to drain first) and exclusive locks are acquired on its target tables. If a table cannot be locked after 3 seconds of waiting processing moves to the next event monitor.
  - While a table for an event monitor is being upgraded, new events for that event monitor are lost.
  - The tables are altered to drop columns or add or modify columns to make them up to date. If the '<createNewGroups>yes</createNewGroups>' option is specified any new tables are created.
    - Tables are altered using the ALTER TABLE statement. However the ALTOBJ stored procedure is used if a table contains a BLOB or CLOB column with an inline lob length and that inline length needs to decrease or if a VARCHAR column changes to CLOB. The **DB2\_UTIL\_MSGPATH** registry variable can be used to redirect any ALTOBJ messages into a directory other than instanceName/tmp.
    - If a table column is altered or dropped all dependent objects will be invalidated. When these dependent objects are revalidated will be affected by the **auto\_reval** database configuration parameter setting. See the documentation for more details about **auto\_reval**.
  - After a table is altered the ADMIN\_REVALIDATE\_DB\_OBJECTS stored procedure is called to perform a table reorg (if necessary) and ensure the table is usable.
  - The exclusive table lock(s) are released.

- If all the tables for an event monitor were successfully upgraded the VERSIONNUMBER column in SYSCAT.EVENTMONITORS is updated to the current version number and a COMMIT is issued. Otherwise the unit of work is rolled back.
- If an event monitor had been deactivated by the stored procedure, it will be reactivated after the stored procedure has completed processing.
- If the INCLUDES clause was specified on the CREATE EVENT MONITOR statement when an event monitor table was created it is skipped. Check the TABOPTIONS column of SYSCAT.EVENTTABLES to determine if INCLUDES was specified. Note that if an event monitor was created before Version 10.1 and utilized INCLUDES this information is lost and the TABOPTIONS column will contain a blank.
- If the EXCLUDES clause was specified on the CREATE EVENT MONITOR statement when an event monitor table was created it is a candidate for upgrading. The table will be modified to accommodate changes since the event monitor was created. Check the TABOPTIONS column of SYSCAT.EVENTTABLES to determine if EXCLUDES was specified. Note that if an event monitor was created before Version 10.1 and utilized EXCLUDES this information is lost and the TABOPTIONS column will contain a blank.
- The procedure will not return any errors encountered while processing the individual event monitor target tables.
- If a result set is to be returned (see “Procedure parameters” on page 475), keep the following information in mind:
  - A user temporary table space must exist before you run the EVMON\_UPGRADE\_TABLES procedure.
  - The table used for the result set is named SESSION.EVMON\_UPGRADE\_TABLES\_RESULTSET.
  - If SESSION.EVMON\_UPGRADE\_TABLES\_RESULTSET already exists from a previous execution of EVMON\_UPGRADE\_TABLES in the same session, the new results will be appended to the existing table. If the table definition has changed in any way since the existing table was created, it is dropped, and a new version of the table is created.
  - SESSION.EVMON\_UPGRADE\_TABLES\_RESULTSET is dropped automatically when with the connection over which EVMON\_UPGRADE\_TABLES runs is dropped. The user temporary table space used for the table cannot be dropped until the SESSION.EVMON\_UPGRADE\_TABLES\_RESULTSET table is dropped.
- During an online fix pack update, a Db2 pureScale instance must be in homogeneous state before calling this procedure. If the instance is in heterogeneous state, this procedure cannot update the monitor event tables. For more details, see “Database and instance operations affected by an online fix pack update in progress” in *Installing Db2 Servers*.

## Example

A user created the following event monitors in Db2 Version 9.7:

```
create event monitor lock for locking write to unformatted event table
create event monitor act for activities write to table control (in
actspace), activity (in actspace), activitystmt (in actspace), activityvals
(in actspace)
create event monitor stat for statistics write to table
create event monitor conn for connections write to table
```

After upgrading the database to the current release they upgrade all the event monitor tables using the following command:

```
call evmon_upgrade_tables(null, null, null, ?, ?, ?)
```

If instead they only wanted to upgrade act, they could use this command:

```
call evmon_upgrade_tables('ACT', null, null, ?, ?)
```

Alternatively they could choose to upgrade only the activities event monitors by using this command:

```
call evmon_upgrade_tables(null, 'ACTIVITIES', null, ?, ?, ?)
```

## Information Returned

Table 92 describes the information returned in the result set if you choose to have it created.

*Table 92. Information returned by EVMON\_UPGRADE\_TABLES to table SESSION.EVMON\_UPGRADE\_TABLES\_RESULTSET*

Column name	Data type	Description or corresponding monitor element
EVMON_NAME	VARCHAR(128)	Name of the event monitor for which this diagnostic message applies.
EVMON_TYPE	VARCHAR(128)	Event type of the event monitor for which this diagnostic message applies.
MESSAGE_TIME	TIMESTAMP	message_time - Timestamp Control Table Message monitor element
MESSAGE_TEXT	VARCHAR(1024)	Diagnostic message.

## MON\_CAPTURE\_ACTIVITY\_IN\_PROGRESS procedure - Collect activity information for activities event monitor

The MON\_CAPTURE\_ACTIVITY\_IN\_PROGRESS procedure gathers information about a specified activity and writes the information to the active activities event monitor.

When you apply this procedure to an activity with child activities, the procedure recursively generates a record for each child activity. This information is collected and sent when you call the procedure; the procedure does not wait until the parent activity completes execution. The record of the activity in the event monitor is marked as a partial record.

## Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

## Default PUBLIC privilege

None

## Syntax

```
►►—MON_CAPTURE_ACTIVITY_IN_PROGRESS—(—application_handle—,—————►
►—uow_id—,—activity_id—)—————►◄
```

The schema is SYSPROC.

## Procedure parameters

If you do not specify all of the following parameters, no activity is found, and an SQL0171N error with SQLSTATE 42815 is returned.

### *application\_handle*

An input argument of type BIGINT that specifies the handle of the application whose activity information is to be captured.

### *uow\_id*

An input argument of type INTEGER that specifies the unit of work ID of the activity whose information is to be captured.

### *activity\_id*

An input argument of type INTEGER that specifies the activity ID that uniquely identifies the activity within the unit of work whose information is to be captured.

## Usage notes

- If there is no active activities event monitor, an SQL1633W error with SQLSTATE 01H53 is returned.
- Activity information is collected only on the coordinator member for the activity.

## Example

Assume that a user complains that stored procedure MYSCHEMA.MYSLOWSTP seems to be running slower than usual. The administrator wants to investigate the cause of the slowdown. Investigating while the stored procedure is running is not practical, so the administrator decides to capture information about the stored procedure activity and all of the activities nested within it.

An event monitor for database activities named DB2ACTIVITIES has been activated. The administrator uses the MON\_GET\_ACTIVITY function to obtain the application handle, unit of work ID, and activity ID for the call of this stored procedure. Assuming that the activity is identified by an application handle of 1, a unit of work ID of 2, and an activity ID of 3, the administrator can now issue the call to MON\_CAPTURE\_ACTIVITY\_IN\_PROGRESS as follows:

```
CALL MON_CAPTURE_ACTIVITY_IN_PROGRESS(1,2,3)
```

After the procedure is completed, the administrator can use the following table function to find out where the activity spent its time. The function retrieves the information from the DB2ACTIVITIES event monitor.

```
CREATE FUNCTION SHOWCAPTUREDACTIVITY(APPHNDL BIGINT,
 UOWID INTEGER,
 ACTIVITYID INTEGER)
 RETURNS TABLE (UOW_ID INTEGER, ACTIVITY_ID INTEGER, STMT_TEXT VARCHAR(40),
 LIFE_TIME DOUBLE)
 LANGUAGE SQL
 READS SQL DATA
```

```

NO EXTERNAL ACTION
DETERMINISTIC
RETURN WITH RAH (LEVEL, APPL_ID, PARENT_UOW_ID, PARENT_ACTIVITY_ID,
 UOW_ID, ACTIVITY_ID, STMT_TEXT, ACT_EXEC_TIME) AS
(SELECT 1, ROOT.APPL_ID, ROOT.PARENT_UOW_ID,
 ROOT.PARENT_ACTIVITY_ID, ROOT.UOW_ID, ROOT.ACTIVITY_ID,
 ROOTSTMT.STMT_TEXT, ACT_EXEC_TIME
 FROM ACTIVITY_DB2ACTIVITIES ROOT, ACTIVITYSTMT_DB2ACTIVITIES ROOTSTMT
 WHERE ROOT.APPL_ID = ROOTSTMT.APPL_ID AND ROOT.AGENT_ID = APPHNDL
 AND ROOT.UOW_ID = ROOTSTMT.UOW_ID AND ROOT.UOW_ID = UOWID
 AND ROOT.ACTIVITY_ID = ROOTSTMT.ACTIVITY_ID AND ROOT.ACTIVITY_ID = ACTIVITYID
 UNION ALL
 SELECT PARENT.LEVEL +1, CHILD.APPL_ID, CHILD.PARENT_UOW_ID,
 CHILD.PARENT_ACTIVITY_ID, CHILD.UOW_ID,
 CHILD.ACTIVITY_ID, CHILDSTMT.STMT_TEXT, CHILD.ACT_EXEC_TIME
 FROM RAH PARENT, ACTIVITY_DB2ACTIVITIES CHILD,
 ACTIVITYSTMT_DB2ACTIVITIES CHILDSTMT
 WHERE PARENT.APPL_ID = CHILD.APPL_ID AND
 CHILD.APPL_ID = CHILDSTMT.APPL_ID AND
 PARENT.UOW_ID = CHILD.PARENT_UOW_ID AND
 CHILD.UOW_ID = CHILDSTMT.UOW_ID AND
 PARENT.ACTIVITY_ID = CHILD.PARENT_ACTIVITY_ID AND
 CHILD.ACTIVITY_ID = CHILDSTMT.ACTIVITY_ID AND
 PARENT.LEVEL < 64
)
 SELECT UOW_ID, ACTIVITY_ID, SUBSTR(STMT_TEXT,1,40),
 ACT_EXEC_TIME AS
 LIFE_TIME
 FROM RAH

```

The following sample query uses the table function:

```

SELECT * FROM TABLE(SHOWCAPTUREDACTIVITY(1, 2, 3))
 AS ACTS ORDER BY UOW_ID, ACTIVITY_ID

```

## **MON\_COLLECT\_STATS procedure - Collect and reset statistics for workload management objects and operating system resources**

The MON\_COLLECT\_STATS procedure gathers or resets statistics for service classes, workloads, work classes, threshold queues, and operating system resources.

If there is an active statistics event monitor, the MON\_COLLECT\_STATS procedure writes the statistics it collects to the statistics event monitor.

### **Authorization**

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

### **Default PUBLIC privilege**

None

## Syntax

►► MON\_COLLECT\_STATS ( *wait* , *statistics\_timestamp* ) ►►

The schema is SYSPROC.

## Routine parameters

### *wait*

An optional input argument of type CHAR that specifies whether this procedure returns immediately after initiating a statistics collection and reset. If 'Y' is specified, then the procedure will not return until all statistics have been written and flushed to the statistics event monitor tables. Otherwise, the procedure will return immediately after initiating a statistics collection and reset.

### *statistics\_timestamp*

An optional output argument of type TIMESTAMP that returns the timestamp value for the beginning of the statistics collection.

## Usage notes

The MON\_COLLECT\_STATS procedure performs the same collection operation (send statistics to the active statistics event monitor) and reset operation that occur automatically on the interval defined by the **wlm\_collect\_int** database configuration parameter.

If you call the procedure while another collection and reset request is in progress (for example, while another invocation of the procedure is running or automatic collection is occurring), SQL1632W with SQLSTATE 01H53 is returned, and your new request is ignored.

In asynchronous mode, the MON\_COLLECT\_STATS procedure only starts the collection and reset process. The procedure might return to the caller before all statistics have been written to the active statistics event monitor. Depending on how quickly the statistics collection and reset occur, the call to the MON\_COLLECT\_STATS procedure (which is itself an activity) is counted in the statistics for either the prior collection interval or the new collection interval that has just started.

In synchronous mode, the MON\_COLLECT\_STATS procedure does not return until the statistics collection is complete and all statistics are written to the tables of any active statistics event monitors. The timestamp at which the statistics collection began is returned via the *statistics\_timestamp* output parameter.

## Examples

1. Call MON\_COLLECT\_STATS to initiate statistics collection and reset.

```
CALL MON_COLLECT_STATS()
```

The following is an example of output from this query:

```
Return Status = 0
```

2. Call MON\_COLLECT\_STATS to collect and reset statistics, and return only when data has been written to statistics event monitor tables.

```
CALL MON_COLLECT_STATS('Y', ::collect_timestamp)
```

The following is an example of output from this query:

```
Return Status = 0
```

3. Call MON\_COLLECT\_STATS to collect and reset statistics while another call is in progress.

```
CALL MON_COLLECT_STATS()
```

The following is an example of output from this query:

```
SQL1632W The collect and reset statistics request was ignored because
another collect and reset statistics request is already in progress.
```

### **MON\_FORMAT\_LOCK\_NAME - Format the internal lock name and return details**

The MON\_FORMAT\_LOCK\_NAME table function formats the internal lock name and returns details regarding the lock in a row-based format.

Each returned row consists of a *key-value* pair relevant for that particular lock.

To get information about locks, use the MON\_FORMAT\_LOCK\_NAME, MON\_GET\_LOCKS, and, MON\_GET\_APPL\_LOCKWAIT table functions instead of the SNAPLOCKWAIT administrative view and SNAP\_GET\_LOCKWAIT table function, and the SNAPLOCK administrative view and SNAP\_GET\_LOCK table function, which are deprecated in Fixpack 1 of Version 9.7.

### **Authorization**

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### **Default PUBLIC privilege**

None

### **Syntax**

►►—MON\_FORMAT\_LOCK\_NAME—(—*lockname*—)—————►◄

The schema is SYSPROC.

### **Table function parameters**

#### ***lockname***

An input argument of type VARCHAR(32) that specifies the internal binary name of the lock that is to be formatted. A NULL value results in error SQL0171N being returned.

## Information returned

Table 93. Information returned by the MON\_FORMAT\_LOCK\_NAME table function

Column name	Data type	Description
NAME	VARCHAR(256)	Element of the lock name. See following table for more details.
VALUE	VARCHAR(1024)	Value of the element.

Not all elements that make up the specified lock name are returned; only those *key-value* pairs that are relevant are returned.

The elements that can be returned are as follows:

Table 94. Monitor elements that can be returned

Element name	Description	Possible values or monitor element
LOCK_OBJECT_TYPE	lock_object_type - Lock object type waited on monitor element	lock_object_type - Lock object type  For possible values, see “lock_object_type - Lock object type waited on monitor element”.
DATA_PARTITION_ID	data_partition_id - Data partition identifier monitor element	data_partition_id - Data partition identifier
TBSP_NAME	The name of a table space	tablespace_name - Table space name
TABSCHEMA	table_schema - Table schema name monitor element	table_schema - Table schema name
TABNAME	table_name - Table name monitor element	table_name - Table name
ROWID	Row ID of the table	-
PAGEID	The page ID	-
WORKLOAD_NAME	workload_name - Workload name monitor element	workload_name - Workload name
STORAGE_GRP_ID	The storage group ID	-
BUFFERPOOL_NAME	Name of the buffer pool	-
FED_SERVER_NAME	Name of the federation server	-
FED_USER_NAME	Name of the federation user mapping	-

Table 94. Monitor elements that can be returned (continued)

Element name	Description	Possible values or monitor element
<b>SEQ_OPERATION</b>	Operation requesting a sequence lock	Possible values are: <ul style="list-style-type: none"> <li>• AUDIT_DDL</li> <li>• AUTONOMIC_POLICIES</li> <li>• CATALOG_ARRAY</li> <li>• CATALOG_COL_DICT_STATE</li> <li>• DESCRIBE</li> <li>• INIT_AUDIT</li> <li>• INIT_EVMON</li> <li>• INIT_PACKAGE</li> <li>• INIT_ROLE_ID</li> <li>• INIT_ROUTINE_ID</li> <li>• INIT_TRUSTED_CTX</li> <li>• PACKAGE_CREATION</li> <li>• STATIC_STMT</li> <li>• TEMP_TBSPACE</li> <li>• TRUSTED_CTX</li> <li>• USER_TEMP_TBSPACE</li> <li>• VERSION_TIMES</li> <li>• WLM</li> </ul>
<b>CONTAINER_ID</b>	container_id - Container identification monitor element	-
<b>STMT_UID</b>	The statement ID	-
<b>PACKAGE_TOKEN</b>	The package token	-
<b>INTERNAL</b>	Reserved for internal use	-

## Examples

The internal lock name is returned in a variety of situations, such as being written to the **db2diag** log files, or as the value of the **lock\_name** monitor element. The following example shows how to use the **MON\_FORMAT\_LOCK\_NAME** table function to find out further information about the lock, in this case with a lock name of 0000000E000000000000B00C152.

```
SELECT SUBSTR(NAME,1,20) AS NAME,
 SUBSTR(VALUE,1,50) AS VALUE
FROM
 TABLE(MON_FORMAT_LOCK_NAME('0000000E000000000000B00C152')) as LOCK
```

The following output is returned:

```
NAME VALUE

LOCK_OBJECT_TYPE ROW
ROWID 0
DATA_PARTITION_ID 49408
PAGEID 184549376
TBSP_NAME SYSCATSPACE
```

5 record(s) selected.

## MON\_FORMAT\_XML\_COMPONENT\_TIMES\_BY\_ROW - Get formatted row-based component times

The MON\_FORMAT\_XML\_COMPONENT\_TIMES\_BY\_ROW table function returns formatted row-based output for the component times contained in an XML metrics document.

### Syntax

►►—MON\_FORMAT\_XML\_COMPONENT\_TIMES\_BY\_ROW—(—*xml doc*—)—————►►

The schema is SYSPROC.

### Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Routine parameters

#### *xml doc*

An input argument of type BLOB(8M) that contains an XML document with either a system\_metrics or activity\_metrics element. XML documents with these elements can be obtained from the following sources:

- Returned by one of the MON\_GET\_\*\_DETAILS table functions.
- From the metrics column output by statistics and activity event monitors.
- From the formatted output of the unit of work, or package cache event monitors.

### Information returned

Table 95. Information returned for MON\_FORMAT\_XML\_COMPONENT\_TIMES\_BY\_ROW

Column Name	Data Type	Description
METRIC_NAME	VARCHAR(128)	The unique identifier for the total time metric value.
PROC_METRIC_NAME	VARCHAR(128)	The unique identifier for the processing time metric.
TOTAL_TIME_VALUE	BIGINT	The total time value in milliseconds corresponding to metric_name.
PROC_TIME_VALUE	BIGINT	The processing time value in milliseconds corresponding to proc_metric_name
COUNT	BIGINT	Number of occurrences of this type of interval.
PARENT_METRIC_NAME	VARCHAR(128)	The identifier of the parent total time metric whose value contains the total_time_value as a subset
PARENT_PROC_METRIC_NAME	VARCHAR(128)	The identifier of the parent processing time metric whose value contains the proc_time_value as a subset

XML documents that contain an element of type *system\_metrics* are generated from the following interfaces:

- MON\_GET\_CONNECTION\_DETAILS
- MON\_GET\_SERVICE\_SUBCLASS\_DETAILS
- MON\_GET\_UNIT\_OF\_WORK\_DETAILS
- MON\_GET\_WORKLOAD\_DETAILS
- DETAILS\_XML column from a STATISTICS event monitor
- METRICS column produced by EVMON\_FORMAT\_UE\_TO\_TABLES for the UNIT OF WORK event monitor
- XMLREPORT column of EVMON\_FORMAT\_UE\_TO\_XML for the UNIT OF WORK event monitor

See Table 96 for the types of metrics and their parent metrics that are returned from the XML document in this case:

*Table 96. Metric names returned by MON\_FORMAT\_XML\_COMPONENT\_TIMES\_BY\_ROW for XML documents containing a system\_metrics element type*

Metric Name	Proc Metric Name	Parent Metric Name	Parent Proc Metric Name	Description of metric or Monitor element
TOTAL_BACKUP_TIME	TOTAL_BACKUP_PROC_TIME	TOTAL_RQST_TIME	TOTAL_RQST_TIME	total_backup_time - Total elapsed time for doing online backups monitor element
TOTAL_COL_SYNOPSIS_TIME	TOTAL_COL_SYNOPSIS_PROC_TIME	TOTAL_SECTION_TIME	TOTAL_COL_PROG_TIME	total_col_synopsis_time - Total column-organized synopsis table monitor element
TOTAL_COL_TIME	TOTAL_COL_PROC_TIME	TOTAL_SECTION_TIME	TOTAL_SECTION_PROC_TIME	total_col_time - Total column-oriented time
TOTAL_COMMIT_TIME	TOTAL_COMMIT_PROC_TIME	TOTAL_RQST_TIME	TOTAL_RQST_TIME	total_commit_time - Total commit time
TOTAL_COMPILE_TIME	TOTAL_COMPILE_PROC_TIME	TOTAL_RQST_TIME	TOTAL_RQST_TIME	total_compile_time - Total compile time
TOTAL_CONNECT_AUTHENTICATION_TIME	TOTAL_CONNECT_AUTHENTICATION_PROC_TIME	TOTAL_CONNECT_REQUEST_TIME	TOTAL_CONNECT_REQUEST_PROC_TIME	total_connect_authentication_time - Total connection or switch user authentication request time
TOTAL_CONNECT_REQUEST_TIME	TOTAL_CONNECT_REQUEST_PROC_TIME	TOTAL_RQST_TIME	TOTAL_RQST_TIME	total_connect_request_time - Total connection or switch user request time
TOTAL_IMPLICIT_COMPILE_TIME	TOTAL_IMPLICIT_COMPILE_PROC_TIME	TOTAL_RQST_TIME	TOTAL_RQST_TIME	total_implicit_compile_time - Total implicit compile time
TOTAL_INDEX_BUILD_TIME	TOTAL_INDEX_BUILD_PROC_TIME	TOTAL_SECTION_TIME	TOTAL_SECTION_PROC_TIME	total_index_build_time - Total time spent building indexes due to index creation or recreate/rebuild monitor element
TOTAL_LOAD_TIME	TOTAL_LOAD_PROC_TIME	TOTAL_RQST_TIME	TOTAL_RQST_TIME	total_load_time - Total load time
TOTAL_REORG_TIME	TOTAL_REORG_PROC_TIME	TOTAL_RQST_TIME	TOTAL_RQST_TIME	total_reorg_time - Total reorganization time
TOTAL_ROLLBACK_TIME	TOTAL_ROLLBACK_PROC_TIME	TOTAL_RQST_TIME	TOTAL_RQST_TIME	total_rollback_time - Total rollback time

Table 96. Metric names returned by MON\_FORMAT\_XML\_COMPONENT\_TIMES\_BY\_ROW for XML documents containing a system\_metrics element type (continued)

Metric Name	Proc Metric Name	Parent Metric Name	Parent Proc Metric Name	Description of metric or Monitor element
TOTAL_ROUTINE_USER_CODE_TIME	TOTAL_ROUTINE_USER_CODE_PROC_TIME	TOTAL_RQST_TIME	TOTAL_RQST_TIME	total_routine_user_code_time - Total routine user code time
TOTAL_RQST_TIME	NULL	NULL	NULL	total_rqst_time - Total request time
TOTAL_RUNSTATS_TIME	TOTAL_RUNSTATS_PROC_TIME	TOTAL_RQST_TIME	TOTAL_RQST_TIME	total_runstats_time - Total runtime statistics
TOTAL_SECTION_SORT_TIME	TOTAL_SECTION_SORT_PROC_TIME	TOTAL_SECTION_TIME	TOTAL_SECTION_PROC_TIME	total_section_sort_time - Total section sort time
TOTAL_SECTION_TIME	TOTAL_SECTION_PROC_TIME	TOTAL_RQST_TIME	TOTAL_RQST_TIME	total_section_time - Total section time
TOTAL_STATS_FABRICATION_TIME	TOTAL_STATS_FABRICATION_PROC_TIME	TOTAL_COMPILE_TIME	TOTAL_COMPILE_PROC_TIME	total_stats_fabrication_time - Total statistics fabrication time
TOTAL_SYNC_RUNSTATS_TIME	TOTAL_SYNC_RUNSTATS_PROC_TIME	TOTAL_COMPILE_TIME	TOTAL_COMPILE_PROC_TIME	total_sync_runstats_time - Total synchronous runstats time

XML documents that contain an element of type *activity\_metrics* are generated from the following interfaces:

- MON\_GET\_ACTIVITY\_DETAILS
- MON\_GET\_PKG\_CACHE\_STMT\_DETAILS
- DETAILS\_XML column from an ACTIVITY event monitor
- METRICS column produced by EVMON\_FORMAT\_UE\_TO\_TABLES for the PACKAGE CACHE event monitor
- XMLREPORT column of EVMON\_FORMAT\_UE\_TO\_XML for the PACKAGE CACHE event monitor

See Table 97 for the types of metrics and their parent metrics that are returned from the XML document in this case:

Table 97. Metric names returned by MON\_FORMAT\_XML\_COMPONENT\_TIMES\_BY\_ROW for XML documents containing an activity\_metrics element type

Metric Name	Proc Metric Name	Parent Metric Name	Parent Proc Metric Name	Description or Monitor element
STMT_EXEC_TIME	NULL	NULL	NULL	stmt_exec_time - Statement execution time
TOTAL_COL_TIME	TOTAL_COL_PROC_TIME	TOTAL_SECTION_TIME	TOTAL_SECTION_PROC_TIME	total_col_time - Total column-oriented time
TOTAL_INDEX_BUILD_TIME	TOTAL_INDEX_BUILD_PROC_TIME	TOTAL_SECTION_TIME	TOTAL_SECTION_PROC_TIME	total_index_build_time - Total time spent building indexes due to index creation or recreate/rebuild monitor element
TOTAL_ROUTINE_NON_SECT_TIME	TOTAL_ROUTINE_NON_SECT_PROC_TIME	TOTAL_ROUTINE_TIME	STMT_EXEC_TIME	total_routine_non_sect_time - Non-section routine execution time

Table 97. Metric names returned by MON\_FORMAT\_XML\_COMPONENT\_TIMES\_BY\_ROW for XML documents containing an activity\_metrics element type (continued)

Metric Name	Proc Metric Name	Parent Metric Name	Parent Proc Metric Name	Description or Monitor element
TOTAL_ROUTINE_TIME	NULL	STMT_EXEC_TIME	NULL	total_routine_time - Total routine time
TOTAL_ROUTINE_USER_CODE_TIME	TOTAL_ROUTINE_USER_CODE_PROC_TIME	TOTAL_ROUTINE_NON_SECT_TIME	TOTAL_ROUTINE_NON_SECT_PROC_TIME	total_routine_user_code_time - Total routine user code time
TOTAL_SECTION_SORT_TIME	TOTAL_SECTION_SORT_PROC_TIME	TOTAL_SECTION_TIME	TOTAL_SECTION_PROC_TIME	total_section_sort_time - Total section sort time
TOTAL_SECTION_TIME	TOTAL_SECTION_PROC_TIME	STMT_EXEC_TIME	STMT_EXEC_TIME	total_section_time - Total section time

### Example

The following example returns the breakdown of component times within the database manager for service subclasses, which shows both the total time spent in any given component, as well as the amount of time that was actually spent processing, rather than waiting, in a component.

```
SELECT SUBSTR(T.SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS,
 SUBSTR(T.SERVICE_SUBCLASS_NAME,1,19) AS SUBCLASS,
 T.MEMBER,
 SUBSTR(COMP.METRIC_NAME,1,15) AS METRIC_NAME
 SUBSTR(COMP.PARENT_METRIC_NAME,1,15) AS PARENT_NAME
 COMP.TOTAL_TIME_VALUE AS TOTAL_TIME,
 COMP.PROC_TIME_VALUE AS TOTAL_PROC_TIME,
 COMP.COUNT
FROM TABLE (MON_GET_SERVICE_SUBCLASS_DETAILS(NULL,
 NULL,-2)) AS T,
 TABLE(MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW(
 T.DETAILS
)) AS COMP

WHERE COMP.PARENT_METRIC_NAME IS NOT NULL;
```

The following is an example of output from this query.

SUPERCLASS	SUBCLASS	MEMBER	METRIC_NAME	PARENT_NAME	...
-----	-----	-----	-----	-----	-----
MYSC	MYSSC	0	TOTAL_COMPILE_T	TOTAL_RQST_TIME...	
MYSC	MYSSC	0	TOTAL_IMPLICIT	TOTAL_RQST_TIME...	
MYSC	MYSSC	0	TOTAL_SECTION_T	TOTAL_RQST_TIME...	
MYSC	MYSSC	0	TOTAL_COMMIT_TI	TOTAL_RQST_TIME...	
MYSC	MYSSC	0	TOTAL_ROLLBACK	TOTAL_RQST_TIME...	
MYSC	MYSSC	0	TOTAL_RUNSTATS	TOTAL_RQST_TIME...	
MYSC	MYSSC	0	TOTAL_REORG_TIM	TOTAL_RQST_TIME...	
MYSC	MYSSC	0	TOTAL_LOAD_TIME	TOTAL_RQST_TIME...	
MYSC	MYSSC	0	TOTAL_SECTION_S	TOTAL_SECTION_T...	

9 record(s) selected.

The following is a continuation of sample output from this query.

...TOTAL_TIME	TOTAL_PROC_TIME	COUNT
-----	-----	-----
...	100	100
...	0	0
...	1253	953
...	213	153
...		0

```

... 0 0 0
... 0 0 0
... 0 0 0
... 0 0 0
... 0 0 0

```

9 record(s) selected.

## MON\_FORMAT\_XML\_METRICS\_BY\_ROW - Get formatted row-based output for all metrics

The MON\_FORMAT\_XML\_METRICS\_BY\_ROW table function returns formatted row-based output for all metrics contained in an XML metrics document.

### Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

### Default PUBLIC privilege

None

### Syntax

```

➤➤ MON_FORMAT_XML_METRICS_BY_ROW (—xml doc—) ➤➤

```

The schema is SYSPROC.

### Routine parameters

#### *xml doc*

An input argument of type BLOB(8M) that contains an XML document with either a *system\_metrics* or *activity\_metrics* element. XML documents with these elements can be obtained from the following sources:

- Returned by one of the MON\_GET\_\*\_DETAILS table functions.
- From the metrics column output by statistics and activity event monitors.
- From the formatted output of the unit of work, or package cache event monitors.

### Information returned

Table 98. Information returned for MON\_FORMAT\_XML\_METRICS\_BY\_ROW

Column Name	Data Type	Description
METRIC_NAME	VARCHAR(128)	The unique identifier for the total time metric value.
VALUE	BIGINT	The current value of the metric.

XML documents that contain an element of type *system\_metrics* are generated from the following interfaces:

- MON\_GET\_CONNECTION\_DETAILS
- MON\_GET\_SERVICE\_SUBCLASS\_DETAILS

- MON\_GET\_UNIT\_OF\_WORK\_DETAILS
- MON\_GET\_WORKLOAD\_DETAILS
- DETAILS\_XML column from a STATISTICS event monitor
- METRICS column produced by EVMON\_FORMAT\_UE\_TO\_TABLES for the UNIT OF WORK event monitor
- XMLREPORT column of EVMON\_FORMAT\_UE\_TO\_XML for the UNIT OF WORK event monitor

See Table 99 for the types of metrics that are returned from the XML document in this case:

*Table 99. Metric names returned by MON\_FORMAT\_XML\_METRICS\_BY\_ROW for XML documents containing a system\_metrics element type*

Metric Name	Description of metric or Monitor element
ACT_ABORTED_TOTAL	act_aborted_total - Total aborted activities
ACT_COMPLETED_TOTAL	act_completed_total - Total completed activities
ACT_REJECTED_TOTAL	act_rejected_total - Total rejected activities
ACT_RQSTS_TOTAL	act_rqsts_total - Total activity requests
AGENT_WAIT_TIME	agent_wait_time - Agent wait time
AGENT_WAITS_TOTAL	agent_waits_total - Total agent waits
APP_ACT_ABORTED_TOTAL	app_act_aborted_total - Total failed external coordinator activities
APP_ACT_COMPLETED_TOTAL	app_act_completed_total - Total successful external coordinator activities
APP_ACT_REJECTED_TOTAL	app_act_rejected_total - Total rejected external coordinator activities
APP_RQSTS_COMPLETED_TOTAL	app_rqsts_completed_total - Total application requests completed
AUDIT_EVENTS_TOTAL	audit_events_total - Total audit events
AUDIT_FILE_WRITE_WAIT_TIME	audit_file_write_wait_time - Audit file write wait time
AUDIT_FILE_WRITES_TOTAL	audit_file_writes_total - Total audit files written
AUDIT_SUBSYSTEM_WAIT_TIME	audit_subsystem_wait_time - Audit subsystem wait time
AUDIT_SUBSYSTEM_WAITS_TOTAL	audit_subsystem_waits_total - Total audit subsystem waits
CAT_CACHE_INSERTS	cat_cache_inserts - Catalog cache inserts
CAT_CACHE_LOOKUPS	cat_cache_lookups - Catalog cache lookups
CF_WAIT_TIME	cf_wait_time - cluster caching facility wait time
CF_WAITS	cf_waits - Number of cluster caching facility waits
CLIENT_IDLE_WAIT_TIME	client_idle_wait_time - Client idle wait time
COL_VECTOR_CONSUMER_OVERFLOW	col_vector_consumer_overflows - Columnar vector consumer overflows monitor element.
COMM_EXIT_WAIT_TIME	comm_exit_wait_time - Communication buffer exit wait time monitor element
COMM_EXIT_WAITS	comm_exit_waits - Communication buffer exit number of waits monitor element
DEADLOCKS	deadlocks - Deadlocks detected
DIAGLOG_WRITE_WAIT_TIME	diaglog_write_wait_time - Diagnostic log file write wait time
DIAGLOG_WRITES_TOTAL	diaglog_writes_total - Total diagnostic log file writes
DIRECT_READ_REQS	direct_read_reqs - Direct read requests
DIRECT_READ_TIME	direct_read_time - Direct read time

Table 99. Metric names returned by `MON_FORMAT_XML_METRICS_BY_ROW` for XML documents containing a `system_metrics` element type (continued)

Metric Name	Description of metric or Monitor element
DIRECT_READS	direct_reads - Direct reads from database
DIRECT_WRITE_REQS	direct_write_reqs - Direct write requests
DIRECT_WRITE_TIME	direct_write_time - Direct write time
DIRECT_WRITES	direct_writes - Direct writes to database
DISABLED_PEDS	disabled_peds - Disabled partial early distincts
EVMON_WAIT_TIME	evmon_wait_time - Event monitor wait time
EVMON_WAITS_TOTAL	evmon_waits_total - Event monitor total waits
FCM_MESSAGE_RECV_VOLUME	fcm_message_recv_volume - FCM message received volume
FCM_MESSAGE_RECV_WAIT_TIME	fcm_message_recv_wait_time - FCM message received wait time
FCM_MESSAGE_RECVS_TOTAL	fcm_message_recv_total - Total FCM message receives
FCM_MESSAGE_SEND_VOLUME	fcm_message_send_volume - FCM message send volume
FCM_MESSAGE_SEND_WAIT_TIME	fcm_message_send_wait_time - FCM message send wait time
FCM_MESSAGE_SENDS_TOTAL	fcm_message_sends_total - Total FCM message sends
FCM_RECV_VOLUME	fcm_recv_volume - FCM received volume
FCM_RECV_WAIT_TIME	fcm_recv_wait_time - FCM received wait time
FCM_RECVS_TOTAL	fcm_recv_total - FCM receives total
FCM_SEND_VOLUME	fcm_send_volume - FCM send volume
FCM_SEND_WAIT_TIME	fcm_send_wait_time - FCM send wait time
FCM_SENDS_TOTAL	fcm_sends_total - FCM sends total
FCM_TQ_RECV_VOLUME	fcm_tq_recv_volume - FCM table queue received volume
FCM_TQ_RECV_WAIT_TIME	fcm_tq_recv_wait_time - FCM table queue received wait time
FCM_TQ_RECVS_TOTAL	fcm_tq_recv_total - FCM table queue receives total
FCM_TQ_SEND_VOLUME	fcm_tq_send_volume - FCM table queue send volume
FCM_TQ_SEND_WAIT_TIME	fcm_tq_send_wait_time - FCM table queue send wait time
FCM_TQ_SENDS_TOTAL	fcm_tq_sends_total - FCM table queue send total
HASH_GRPBY_OVERFLOWS	hash_grpby_overflows - Hash group by overflows
IDA_RECV_VOLUME	The total volume of data the database server received from an in-database analytics process. The value is reported in bytes.
IDA_RECV_WAIT_TIME	The total amount of time spent waiting to receive data from an in-database analytics process.
IDA_RECVS_TOTAL	The total number of times data was received from an in-database analytics process.
IDA_SEND_VOLUME	The total volume of data sent from the database server to an in-database analytics process. The value is reported in bytes.
IDA_SEND_WAIT_TIME	The total amount of time spent waiting to send data to an in-database analytics process.
IDA_SENDS_TOTAL	The total number of times data was sent to an in-database analytics process.
INT_COMMITS	int_commits - Internal commits
INT_ROLLBACKS	int_rollback - Internal rollbacks
IPC_RECV_VOLUME	ipc_recv_volume - Interprocess communication received volume

Table 99. Metric names returned by MON\_FORMAT\_XML\_METRICS\_BY\_ROW for XML documents containing a system\_metrics element type (continued)

Metric Name	Description of metric or Monitor element
IPC_RECV_WAIT_TIME	ipc_recv_wait_time - Interprocess communication received wait time
IPC_RECVS_TOTAL	ipc_recv_total - Interprocess communication receives total
IPC_SEND_VOLUME	ipc_send_volume - Interprocess communication send volume
IPC_SEND_WAIT_TIME	ipc_send_wait_time - Interprocess communication send wait time
IPC_SENDS_TOTAL	ipc_sends_total - Interprocess communication send total
LOCK_ESCALS_GLOBAL	lock_escals_global - Number of global lock escalations
LOCK_ESCALS_LOCKLIST	lock_escals_locklist - Number of locklist lock escalations
LOCK_ESCALS_MAXLOCKS	lock_escals_maxlocks - Number of maxlocks lock escalations
LOCK_ESCALS	lock_escals - Number of lock escalations
LOCK_TIMEOUTS_GLOBAL	lock_timeouts_global - Lock timeouts global
LOCK_TIMEOUTS	lock_timeouts - Number of lock timeouts
LOCK_WAIT_TIME_GLOBAL	lock_wait_time_global - Lock wait time global
LOCK_WAIT_TIME	lock_wait_time - Time waited on locks
LOCK_WAITS_GLOBAL	lock_waits_global - Lock waits global
LOCK_WAITS	lock_waits - Lock waits
LOG_BUFFER_WAIT_TIME	log_buffer_wait_time - Log buffer wait time
LOG_DISK_WAIT_TIME	log_disk_wait_time - Log disk wait time
LOG_DISK_WAITS_TOTAL	log_disk_waits_total - Total log disk waits
NUM_LOG_BUFFER_FULL	num_log_buffer_full - Number of times full log buffer caused agents to wait
NUM_LW_THRESH_EXCEEDED	num_lw_thresh_exceeded - Number of lock wait thresholds exceeded
PKG_CACHE_INSERTS	pkg_cache_inserts - Package cache inserts
PKG_CACHE_LOOKUPS	pkg_cache_lookups - Package cache lookups
POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP	pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool
POOL_COL_GBP_INVALID_PAGES	pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages
POOL_COL_GBP_L_READS	pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads
POOL_COL_GBP_P_READS	pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads
POOL_COL_L_READS	pool_col_l_reads - Buffer pool column-organized logical reads
POOL_COL_LBP_PAGES_FOUND	pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found
POOL_COL_P_READS	pool_col_p_reads - Buffer pool column-organized physical reads
POOL_COL_WRITES	pool_col_writes - Buffer pool column-organized writes
POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP	pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool
POOL_DATA_GBP_INVALID_PAGES	pool_data_gbp_invalid_pages - Group buffer pool invalid data pages
POOL_DATA_GBP_L_READS	pool_data_gbp_l_reads - Group buffer pool data logical reads

Table 99. Metric names returned by `MON_FORMAT_XML_METRICS_BY_ROW` for XML documents containing a `system_metrics` element type (continued)

Metric Name	Description of metric or Monitor element
POOL_DATA_GBP_P_READS	pool_data_gbp_p_reads - Group buffer pool data physical reads
POOL_DATA_L_READS	pool_data_l_reads - Buffer pool data logical reads
POOL_DATA_LBP_PAGES_FOUND	pool_data_lbp_pages_found - Local buffer pool found data pages
POOL_DATA_P_READS	pool_data_p_reads - Buffer pool data physical reads
POOL_DATA_WRITES	pool_data_writes - Buffer pool data writes
POOL_FAILED_ASYNC_COL_REQS	pool_failed_async_col_reqs - Failed column-organized prefetch requests
POOL_FAILED_ASYNC_DATA_REQS	pool_failed_async_data_reqs - Failed data prefetch requests
POOL_FAILED_ASYNC_INDEX_REQS	pool_failed_async_index_reqs - Failed index prefetch requests
POOL_FAILED_ASYNC_OTHER_REQS	pool_failed_async_other_reqs - Failed non-prefetch requests
POOL_FAILED_ASYNC_TEMP_COL_REQS	pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests
POOL_FAILED_ASYNC_TEMP_DATA_REQS	pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces
POOL_FAILED_ASYNC_TEMP_INDEX_REQS	pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces
POOL_FAILED_ASYNC_TEMP_XDA_REQS	pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces
POOL_FAILED_ASYNC_XDA_REQS	pool_failed_async_xda_reqs - Failed XDA prefetch requests
POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP	pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool
POOL_INDEX_GBP_INVALID_PAGES	pool_index_gbp_invalid_pages - Group buffer pool invalid index pages
POOL_INDEX_GBP_L_READS	pool_index_gbp_l_reads - Group buffer pool index logical reads
POOL_INDEX_GBP_P_READS	pool_index_gbp_p_reads - Group buffer pool index physical reads
POOL_INDEX_L_READS	pool_index_l_reads - Buffer pool index logical reads
POOL_INDEX_LBP_PAGES_FOUND	pool_index_lbp_pages_found - Local buffer pool index pages found
POOL_INDEX_P_READS	pool_index_p_reads - Buffer pool index physical reads
POOL_INDEX_WRITES	pool_index_writes - Buffer pool index writes
POOL_QUEUED_ASYNC_COL_PAGES	pool_queued_async_col_pages - Column-organized page prefetch requests
POOL_QUEUED_ASYNC_COL_REQS	pool_queued_async_col_reqs - Column-organized prefetch requests
POOL_QUEUED_ASYNC_DATA_PAGES	pool_queued_async_data_pages - Data pages prefetch requests
POOL_QUEUED_ASYNC_DATA_REQS	pool_queued_async_data_reqs - Data prefetch requests
POOL_QUEUED_ASYNC_INDEX_PAGES	pool_queued_async_index_pages - Index pages prefetch requests
POOL_QUEUED_ASYNC_INDEX_REQS	pool_queued_async_index_reqs - Index prefetch requests
POOL_QUEUED_ASYNC_OTHER_REQS	pool_queued_async_other_reqs - Non-prefetch requests
POOL_QUEUED_ASYNC_TEMP_COL_PAGES	pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests
POOL_QUEUED_ASYNC_TEMP_COL_REQS	pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests

Table 99. Metric names returned by `MON_FORMAT_XML_METRICS_BY_ROW` for XML documents containing a `system_metrics` element type (continued)

Metric Name	Description of metric or Monitor element
POOL_QUEUED_ASYNC_TEMP_DATA_PAGES	pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces
POOL_QUEUED_ASYNC_TEMP_DATA_REQS	pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces
POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES	pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces
POOL_QUEUED_ASYNC_TEMP_INDEX_REQS	pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces
POOL_QUEUED_ASYNC_TEMP_XDA_PAGES	pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces
POOL_QUEUED_ASYNC_TEMP_XDA_REQS	pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces
POOL_QUEUED_ASYNC_XDA_PAGES	pool_queued_async_xda_pages - XDA pages prefetch requests
POOL_QUEUED_ASYNC_XDA_REQS	pool_queued_async_xda_reqs - XDA prefetch requests
POOL_READ_TIME	pool_read_time - Total buffer pool physical read time
POOL_TEMP_COL_L_READS	pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads
POOL_TEMP_COL_P_READS	pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads
POOL_TEMP_DATA_L_READS	pool_temp_data_l_reads - Buffer pool temporary data logical reads
POOL_TEMP_DATA_P_READS	pool_temp_data_p_reads - Buffer pool temporary data physical reads
POOL_TEMP_INDEX_L_READS	pool_temp_index_l_reads - Buffer pool temporary index logical reads
POOL_TEMP_INDEX_P_READS	pool_temp_index_p_reads - Buffer pool temporary index physical reads
POOL_TEMP_XDA_L_READS	pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads
POOL_TEMP_XDA_P_READS	pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads
POOL_WRITE_TIME	pool_write_time - Total buffer pool physical write time
POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP	pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool
POOL_XDA_GBP_INVALID_PAGES	pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages
POOL_XDA_GBP_L_READS	pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests
POOL_XDA_GBP_P_READS	pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests
POOL_XDA_L_READS	pool_xda_l_reads - Buffer pool XDA data logical reads
POOL_XDA_LBP_PAGES_FOUND	pool_xda_lbp_pages_found - Local buffer pool XDA data pages found
POOL_XDA_P_READS	pool_xda_p_reads - Buffer pool XDA data physical reads
POOL_XDA_WRITES	pool_xda_writes - Buffer pool XDA data writes
POST_SHRTHRESHOLD_SORTS	post_shrthreshold_sorts - Post shared threshold sorts

Table 99. Metric names returned by MON\_FORMAT\_XML\_METRICS\_BY\_ROW for XML documents containing a system\_metrics element type (continued)

Metric Name	Description of metric or Monitor element
POST_THRESHOLD_COL_VECTOR_CONSUMERS	post_threshold_col_vector_consumers monitor element
POST_THRESHOLD_HASH_GRPBYS	post_threshold_hash_grpbys - Hash group by threshold
POST_THRESHOLD_PEAS	post_threshold_peas - Partial early aggregation threshold
POST_THRESHOLD_PEDS	post_threshold_peds - Partial early distincts threshold
POST_THRESHOLD_SORTS	post_threshold_sorts - Post threshold sorts
PREFETCH_WAIT_TIME	prefetch_wait_time - Time waited for prefetch
PREFETCH_WAITS	prefetch_waits - Prefetcher wait count
RECLAIM_WAIT_TIME	reclaim_wait_time - Reclaim wait time
ROWS_MODIFIED	rows_modified - Rows modified
ROWS_READ	rows_read - Rows read
ROWS_RETURNED	rows_returned - Rows returned
RQSTS_COMPLETED_TOTAL	rqsts_completed_total - Total requests completed
SORT_OVERFLOWS	sort_overflows - Sort overflows
SPACEMAPPAGE_RECLAIM_WAIT_TIME	spacemappage_reclaim_wait_time - Space map page reclaim wait time
TCPIP_RECV_VOLUME	tcpip_recv_volume - TCP/IP received volume
TCPIP_RECV_WAIT_TIME	tcpip_recv_wait_time - TCP/IP received wait time
TCPIP_RECVS_TOTAL	tcpip_recv_total - TCP/IP receives total
TCPIP_SEND_VOLUME	tcpip_send_volume - TCP/IP send volume
TCPIP_SEND_WAIT_TIME	tcpip_send_wait_time - TCP/IP send wait time
TCPIP_SENDS_TOTAL	tcpip_sends_total - TCP/IP sends total
THRESH_VIOLATIONS	thresh_violations - Number of threshold violations
TOTAL_ACT_TIME	total_act_time - Total activity time
TOTAL_ACT_WAIT_TIME	total_act_wait_time - Total activity wait time
TOTAL_APP_COMMITS	total_app_commits - Total application commits
TOTAL_APP_ROLLBACKS	total_app_rollbacks - Total application rollbacks
TOTAL_APP_RQST_TIME	total_app_rqst_time - Total application request time
TOTAL_APP_SECTION_EXECUTIONS	total_app_section_executions - Total application section executions
TOTAL_BACKUP_PROC_TIME	total_backup_proc_time monitor element
TOTAL_BACKUP_TIME	total_backup_time monitor element
TOTAL_BACKUPS	total_backups monitor element
TOTAL_COL_EXECUTIONS	total_col_executions - Total column-organized executions
TOTAL_COL_PROC_TIME	total_col_proc_time - Total column-organized processing time
TOTAL_COL_TIME	total_col_time - Total column-organized time
TOTAL_COL_VECTOR_CONSUMERS	total_col_vector_consumers monitor element
TOTAL_COMMIT_PROC_TIME	total_commit_proc_time - Total commits processing time
TOTAL_COMMIT_TIME	total_commit_time - Total commit time
TOTAL_COMPILE_COMPILATIONS	total_compilations - Total compilations
TOTAL_COMPILE_PROC_TIME	total_compile_proc_time - Total compile processing time

Table 99. Metric names returned by MON\_FORMAT\_XML\_METRICS\_BY\_ROW for XML documents containing a system\_metrics element type (continued)

Metric Name	Description of metric or Monitor element
TOTAL_COMPILE_TIME	total_compile_time - Total compile time
TOTAL_CONNECT_AUTHENTICATION_PROC_TIME	total_connect_authentication_proc_time - Total connection authentication processing time
TOTAL_CONNECT_AUTHENTICATION_TIME	total_connect_authentication_time - Total connection or switch user authentication request time
TOTAL_CONNECT_AUTHENTIFICATIONS	total_connect_authentications - Connections or switch user authentications performed
TOTAL_CONNECT_REQUEST_PROC_TIME	total_connect_request_proc_time - Total connection or switch user request processing time
TOTAL_CONNECT_REQUEST_TIME	total_connect_request_time - Total connection or switch user request time
TOTAL_CONNECT_REQUESTS	total_connect_requests - Connection or switch user requests
TOTAL_CPU_TIME	total_cpu_time - Total CPU time
TOTAL_DISP_RUN_QUEUE_TIME	total_disp_run_queue_time - Total dispatcher run queue time
TOTAL_EXTENDED_LATCH_WAIT_TIME	total_extended_latch_wait_time - Total extended latch wait time
TOTAL_EXTENDED_LATCH_WAITS	total_extended_latch_waits - Total extended latch waits
TOTAL_HASH_GRPBYS	total_hash_grpbys - Total hash group by operations
TOTAL_IMPLICIT_COMPILATIONS	total_implicit_compilations - Total implicit complications
TOTAL_IMPLICIT_COMPILE_PROC_TIME	total_implicit_compile_proc_time - Total implicit compile processing time
TOTAL_IMPLICIT_COMPILE_TIME	total_implicit_compile_time - Total implicit compile time
TOTAL_INDEX_BUILD_TIME	total_index_build_time monitor element
TOTAL_INDEX_BUILD_PROC_TIME	total_index_build_proc_time monitor element
TOTAL_INDEXES_BUILT	total_indexes_built monitor element
TOTAL_LOAD_PROC_TIME	total_load_proc_time - Total load processing time
TOTAL_LOAD_TIME	total_load_time - Total load time
TOTAL_LOADS	total_loads - Total loads
TOTAL_PEAS	total_peas - Total partial early aggregations
TOTAL_PEDS	total_peds - Total partial early distincts
TOTAL_REORG_PROC_TIME	total_reorg_proc_time - Total reorganization processing time
TOTAL_REORG_TIME	total_reorg_time - Total reorganization time
TOTAL_REORGS	total_reorgs - Total reorganizations
TOTAL_ROLLBACK_PROC_TIME	total_rollback_proc_time - Total rollback processing time
TOTAL_ROLLBACK_TIME	total_rollback_time - Total rollback time
TOTAL_ROUTINE_INVOCATIONS	total_routine_invocations - Total routine invocations
TOTAL_ROUTINE_TIME	total_routine_time - Total routine time
TOTAL_ROUTINE_USER_CODE_PROC_TIME	total_routine_user_code_proc_time - Total routine user code processing time
TOTAL_ROUTINE_USER_CODE_TIME	total_routine_user_code_time - Total routine user code time
TOTAL_RQST_TIME	total_rqst_time - Total request time

Table 99. Metric names returned by `MON_FORMAT_XML_METRICS_BY_ROW` for XML documents containing a `system_metrics` element type (continued)

Metric Name	Description of metric or Monitor element
TOTAL_RUNSTATS_PROC_TIME	total_runstats_proc_time - Total runtime statistics processing time
TOTAL_RUNSTATS_TIME	total_runstats_time - Total runtime statistics time
TOTAL_RUNSTATS	total_runstats - Total runtime statistics
TOTAL_SECTION_PROC_TIME	total_section_proc_time - Total section processing time
TOTAL_SECTION_SORT_PROC_TIME	total_section_sort_proc_time - Total section sort processing time
TOTAL_SECTION_SORT_TIME	total_section_sort_time - Total section sort time
TOTAL_SECTION_SORTS	total_section_sorts - Total section sorts
TOTAL_SECTION_TIME	total_section_time - Total section time
TOTAL_SORTS	total_sorts - Total sorts
TOTAL_STATS_FABRICATION_PROC_TIME	total_stats_fabrication_proc_time - Total statistics fabrication processing time
TOTAL_STATS_FABRICATION_TIME	total_stats_fabrication_time - Total statistics fabrication time
TOTAL_STATS_FABRICATIONS	total_stats_fabrications - Total statistics fabrications
TOTAL_SYNC_RUNSTATS_PROC_TIME	total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time
TOTAL_SYNC_RUNSTATS_TIME	total_sync_runstats_time - Total synchronous RUNSTATS time
TOTAL_SYNC_RUNSTATS	total_sync_runstats - Total synchronous RUNSTATS activities
TOTAL_WAIT_TIME	total_wait_time - Total wait time
TQ_SORT_HEAP_REJECTIONS	tq_sort_heap_rejections - Table queue sort heap rejections
TQ_SORT_HEAP_REQUESTS	tq_sort_heap_requests - Table queue sort heap requests
TQ_TOT_SEND_SPILLS	tq_tot_send_spills - Total number of table queue buffers overflowed
WLM_QUEUE_ASSIGNMENTS_TOTAL	wlm_queue_assignments_total - Workload manager total queue assignments
WLM_QUEUE_TIME_TOTAL	wlm_queue_time_total - Workload manager total queue time

XML documents that contain an element of type `activity_metrics` are generated from the following interfaces:

- `MON_GET_ACTIVITY_DETAILS`
- `MON_GET_PKG_CACHE_STMT_DETAILS`
- `DETAILS_XML` column from an `ACTIVITY` event monitor
- `METRICS` column produced by `EVMON_FORMAT_UE_TO_TABLES` for the `PACKAGE CACHE` event monitor
- `XMLREPORT` column of `EVMON_FORMAT_UE_TO_XML` for the `PACKAGE CACHE` event monitor

See Table 100 for the types of metrics that are returned from the XML document in this case:

Table 100. Metric names returned by `MON_FORMAT_XML_METRICS_BY_ROW` for XML documents containing an `activity_metrics` element type

Metric Name	Description or Monitor element
AUDIT_EVENTS_TOTAL	audit_events_total - Total audit events
AUDIT_FILE_WRITE_WAIT_TIME	audit_file_write_wait_time - Audit file write wait time

Table 100. Metric names returned by MON\_FORMAT\_XML\_METRICS\_BY\_ROW for XML documents containing an activity\_metrics element type (continued)

Metric Name	Description or Monitor element
AUDIT_FILE_WRITES_TOTAL	audit_file_writes_total - Total audit files written
AUDIT_SUBSYSTEM_WAIT_TIME	audit_subsystem_wait_time - Audit subsystem wait time
AUDIT_SUBSYSTEM_WAITS_TOTAL	audit_subsystem_waits_total - Total audit subsystem waits
CF_WAIT_TIME	cf_wait_time - cluster caching facility wait time
CF_WAITS	cf_waits - Number of cluster caching facility waits
COORD_STMT_EXEC_TIME	coord_stmt_exec_time - Execution time for statement by coordinator agent
DEADLOCKS	deadlocks - Deadlocks detected
DIAGLOG_WRITE_WAIT_TIME	diaglog_write_wait_time - Diagnostic log file write wait time
DIAGLOG_WRITES_TOTAL	diaglog_writes_total - Total diagnostic log file writes
DIRECT_READ_REQS	direct_read_reqs - Direct read requests
DIRECT_READ_TIME	direct_read_time - Direct read time
DIRECT_READS	direct_reads - Direct reads from database
DIRECT_WRITE_REQS	direct_write_reqs - Direct write requests
DIRECT_WRITE_TIME	direct_write_time - Direct write time
DIRECT_WRITES	direct_writes - Direct writes to database
DISABLED_PEDS	disabled_peds - Disabled partial early distincts
EVMON_WAIT_TIME	evmon_wait_time - Event monitor wait time
EVMON_WAITS_TOTAL	evmon_waits_total - Event monitor total waits
FCM_MESSAGE_RECV_VOLUME	fcm_message_recv_volume - FCM message received volume
FCM_MESSAGE_RECV_WAIT_TIME	fcm_message_recv_wait_time - FCM message received wait time
FCM_MESSAGE_RECVS_TOTAL	fcm_message_recvs_total - Total FCM message receives
FCM_MESSAGE_SEND_VOLUME	fcm_message_send_volume - FCM message send volume
FCM_MESSAGE_SEND_WAIT_TIME	fcm_message_send_wait_time - FCM message send wait time
FCM_MESSAGE_SENDS_TOTAL	fcm_message_sends_total - Total FCM message sends
FCM_RECV_VOLUME	fcm_recv_volume - FCM received volume
FCM_RECV_WAIT_TIME	fcm_recv_wait_time - FCM received wait time
FCM_RECVS_TOTAL	fcm_recvs_total - FCM receives total
FCM_SEND_VOLUME	fcm_send_volume - FCM send volume
FCM_SEND_WAIT_TIME	fcm_send_wait_time - FCM send wait time
FCM_SENDS_TOTAL	fcm_sends_total - FCM sends total
FCM_TQ_RECV_VOLUME	fcm_tq_recv_volume - FCM table queue received volume
FCM_TQ_RECV_WAIT_TIME	fcm_tq_recv_wait_time - FCM table queue received wait time
FCM_TQ_RECVS_TOTAL	fcm_tq_recvs_total - FCM table queue receives total
FCM_TQ_SEND_VOLUME	fcm_tq_send_volume - FCM table queue send volume
FCM_TQ_SEND_WAIT_TIME	fcm_tq_send_wait_time - FCM table queue send wait time
FCM_TQ_SENDS_TOTAL	fcm_tq_sends_total - FCM table queue send total
HASH_GRPBY_OVERFLOWS	hash_grpby_overflows - Hash group by overflows
IDA_RECV_VOLUME	The total volume of data the database server received from an in-database analytics process. The value is reported in bytes.

Table 100. Metric names returned by MON\_FORMAT\_XML\_METRICS\_BY\_ROW for XML documents containing an activity\_metrics element type (continued)

Metric Name	Description or Monitor element
IDA_RECV_WAIT_TIME	The total amount of time spent waiting to receive data from an in-database analytics process.
IDA_RECVS_TOTAL	The total number of times data was received from an in-database analytics process.
IDA_SEND_VOLUME	The total volume of data sent from the database server to an in-database analytics process. The value is reported in bytes.
IDA_SEND_WAIT_TIME	The total amount of time spent waiting to send data to an in-database analytics process.
IDA_SENDS_TOTAL	The total number of times data was sent to an in-database analytics process.
LOCK_ESCALS_GLOBAL	lock_escals_global - Number of global lock escalations
LOCK_ESCALS_LOCKLIST	lock_escals_locklist - Number of locklist lock escalations
LOCK_ESCALS_MAXLOCKS	lock_escals_maxlocks - Number of maxlocks lock escalations
LOCK_ESCALS	lock_escals - Number of lock escalations
LOCK_TIMEOUTS_GLOBAL	lock_timeouts_global - Lock timeouts global
LOCK_TIMEOUTS	lock_timeouts - Number of lock timeouts
LOCK_WAIT_TIME_GLOBAL	lock_wait_time_global - Lock wait time global
LOCK_WAIT_TIME	lock_wait_time - Time waited on locks
LOCK_WAITS_GLOBAL	lock_waits_global - Lock waits global
LOCK_WAITS	lock_waits - Lock waits
LOG_BUFFER_WAIT_TIME	log_buffer_wait_time - Log buffer wait time
LOG_DISK_WAIT_TIME	log_disk_wait_time - Log disk wait time
LOG_DISK_WAITS_TOTAL	log_disk_waits_total - Total log disk waits
NUM_LOG_BUFFER_FULL	num_log_buffer_full - Number of times full log buffer caused agents to wait
NUM_LW_THRESH_EXCEEDED	num_lw_thresh_exceeded - Number of lock wait thresholds exceeded
POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP	pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool
POOL_COL_GBP_INVALID_PAGES	pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages
POOL_COL_GBP_L_READS	pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads
POOL_COL_GBP_P_READS	pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads
POOL_COL_L_READS	pool_col_l_reads - Buffer pool column-organized logical reads
POOL_COL_LBP_PAGES_FOUND	pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found
POOL_COL_P_READS	pool_col_p_reads - Buffer pool column-organized physical reads
POOL_COL_WRITES	pool_col_writes - Buffer pool column-organized writes
POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP	pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool
POOL_DATA_GBP_INVALID_PAGES	pool_data_gbp_invalid_pages - Group buffer pool invalid data pages
POOL_DATA_GBP_L_READS	pool_data_gbp_l_reads - Group buffer pool data logical reads

Table 100. Metric names returned by MON\_FORMAT\_XML\_METRICS\_BY\_ROW for XML documents containing an activity\_metrics element type (continued)

Metric Name	Description or Monitor element
POOL_DATA_GBP_P_READS	pool_data_gbp_p_reads - Group buffer pool data physical reads
POOL_DATA_L_READS	pool_data_l_reads - Buffer pool data logical reads
POOL_DATA_LBP_PAGES_FOUND	pool_data_lbp_pages_found - Local buffer pool found data pages
POOL_DATA_P_READS	pool_data_p_reads - Buffer pool data physical reads
POOL_DATA_WRITES	pool_data_writes - Buffer pool data writes
POOL_FAILED_ASYNC_COL_REQS	pool_failed_async_col_reqs - Failed column-organized prefetch requests
POOL_FAILED_ASYNC_DATA_REQS	pool_failed_async_data_reqs - Failed data prefetch requests
POOL_FAILED_ASYNC_INDEX_REQS	pool_failed_async_index_reqs - Failed index prefetch requests
POOL_FAILED_ASYNC_OTHER_REQS	pool_failed_async_other_reqs - Failed non-prefetch requests
POOL_FAILED_ASYNC_TEMP_COL_REQS	pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests
POOL_FAILED_ASYNC_TEMP_DATA_REQS	pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces
POOL_FAILED_ASYNC_TEMP_INDEX_REQS	pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces
POOL_FAILED_ASYNC_TEMP_XDA_REQS	pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces
POOL_FAILED_ASYNC_XDA_REQS	pool_failed_async_xda_reqs - Failed XDA prefetch requests
POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP	pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool
POOL_INDEX_GBP_INVALID_PAGES	pool_index_gbp_invalid_pages - Group buffer pool invalid index pages
POOL_INDEX_GBP_L_READS	pool_index_gbp_l_reads - Group buffer pool index logical reads
POOL_INDEX_GBP_P_READS	pool_index_gbp_p_reads - Group buffer pool index physical reads
POOL_INDEX_L_READS	pool_index_l_reads - Buffer pool index logical reads
POOL_INDEX_LBP_PAGES_FOUND	pool_index_lbp_pages_found - Local buffer pool index pages found
POOL_INDEX_P_READS	pool_index_p_reads - Buffer pool index physical reads
POOL_INDEX_WRITES	pool_index_writes - Buffer pool index writes
POOL_QUEUED_ASYNC_COL_PAGES	pool_queued_async_col_pages - Column-organized page prefetch requests
POOL_QUEUED_ASYNC_COL_REQS	pool_queued_async_col_reqs - Column-organized prefetch requests
POOL_QUEUED_ASYNC_DATA_PAGES	pool_queued_async_data_pages - Data pages prefetch requests
POOL_QUEUED_ASYNC_DATA_REQS	pool_queued_async_data_reqs - Data prefetch requests
POOL_QUEUED_ASYNC_INDEX_PAGES	pool_queued_async_index_pages - Index pages prefetch requests
POOL_QUEUED_ASYNC_INDEX_REQS	pool_queued_async_index_reqs - Index prefetch requests
POOL_QUEUED_ASYNC_OTHER_REQS	pool_queued_async_other_reqs - Non-prefetch requests
POOL_QUEUED_ASYNC_TEMP_COL_PAGES	pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests
POOL_QUEUED_ASYNC_TEMP_COL_REQS	pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests
POOL_QUEUED_ASYNC_TEMP_DATA_PAGES	pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces

Table 100. Metric names returned by `MON_FORMAT_XML_METRICS_BY_ROW` for XML documents containing an `activity_metrics` element type (continued)

Metric Name	Description or Monitor element
POOL_QUEUED_ASYNC_TEMP_DATA_REQS	pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces
POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES	pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces
POOL_QUEUED_ASYNC_TEMP_INDEX_REQS	pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces
POOL_QUEUED_ASYNC_TEMP_XDA_PAGES	pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces
POOL_QUEUED_ASYNC_TEMP_XDA_REQS	pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces
POOL_QUEUED_ASYNC_XDA_PAGES	pool_queued_async_xda_pages - XDA pages prefetch requests
POOL_QUEUED_ASYNC_XDA_REQS	pool_queued_async_xda_reqs - XDA prefetch requests
POOL_READ_TIME	pool_read_time - Total buffer pool physical read time
POOL_TEMP_COL_L_READS	pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads
POOL_TEMP_COL_P_READS	pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads
POOL_TEMP_DATA_L_READS	pool_temp_data_l_reads - Buffer pool temporary data logical reads
POOL_TEMP_DATA_P_READS	pool_temp_data_p_reads - Buffer pool temporary data physical reads
POOL_TEMP_INDEX_L_READS	pool_temp_index_l_reads - Buffer pool temporary index logical reads
POOL_TEMP_INDEX_P_READS	pool_temp_index_p_reads - Buffer pool temporary index physical reads
POOL_TEMP_XDA_L_READS	pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads
POOL_TEMP_XDA_P_READS	pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads
POOL_WRITE_TIME	pool_write_time - Total buffer pool physical write time
POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP	pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool
POOL_XDA_GBP_INVALID_PAGES	pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages
POOL_XDA_GBP_L_READS	pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests
POOL_XDA_GBP_P_READS	pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests
POOL_XDA_L_READS	pool_xda_l_reads - Buffer pool XDA data logical reads
POOL_XDA_LBP_PAGES_FOUND	pool_xda_lbp_pages_found - Local buffer pool XDA data pages found
POOL_XDA_P_READS	pool_xda_p_reads - Buffer pool XDA data physical reads
POOL_XDA_WRITES	pool_xda_writes - Buffer pool XDA data writes
POST_SHRTHRESHOLD_SORTS	post_shrthreshold_sorts - Post shared threshold sorts
POST_THRESHOLD_COL_VECTOR_CONSUMERS	post_threshold_col_vector_consumers monitor element
POST_THRESHOLD_HASH_GRPBYS	post_threshold_hash_grpbys - Hash group by threshold
POST_THRESHOLD_PEAS	post_threshold_peas - Partial early aggregation threshold
POST_THRESHOLD_PEDS	post_threshold_peds - Partial early distincts threshold

Table 100. Metric names returned by MON\_FORMAT\_XML\_METRICS\_BY\_ROW for XML documents containing an activity\_metrics element type (continued)

Metric Name	Description or Monitor element
POST_THRESHOLD_SORTS	post_threshold_sorts - Post threshold sorts
PREFETCH_WAIT_TIME	prefetch_wait_time - Time waited for prefetch
PREFETCH_WAITS	prefetch_waits - Prefetcher wait count
RECLAIM_WAIT_TIME	reclaim_wait_time - Reclaim wait time
ROWS_MODIFIED	rows_modified - Rows modified
ROWS_READ	rows_read - Rows read
ROWS_RETURNED	rows_returned - Rows returned
SORT_OVERFLOWS	sort_overflows - Sort overflows
SPACEMAPPAGE_RECLAIM_WAIT_TIME	spacemappage_reclaim_wait_time - Space map page reclaim wait time
STMT_EXEC_TIME	stmt_exec_time - Statement execution time
THRESH_VIOLATIONS	thresh_violations - Number of threshold violations
TOTAL_ACT_TIME	total_act_time - Total activity time
TOTAL_ACT_WAIT_TIME	total_act_wait_time - Total activity wait time
TOTAL_APP_SECTION_EXECUTIONS	total_app_section_executions - Total application section executions
TOTAL_COL_EXECUTIONS	total_col_executions - Total column-organized executions
TOTAL_COL_PROC_TIME	total_col_proc_time - Total column-organized processing time
TOTAL_COL_TIME	total_col_time - Total column-organized time
TOTAL_COL_VECTOR_CONSUMERS	total_col_vector_consumers monitor element
TOTAL_CPU_TIME	total_cpu_time - Total CPU time
TOTAL_DISP_RUN_QUEUE_TIME	total_disp_run_queue_time - Total dispatcher run queue time
TOTAL_EXTENDED_LATCH_WAIT_TIME	total_extended_latch_wait_time - Total extended latch wait time
TOTAL_EXTENDED_LATCH_WAITS	total_extended_latch_waits - Total extended latch waits
TOTAL_HASH_GRPBYS	total_hash_grpbys - Total hash group by operations
TOTAL_INDEX_BUILD_TIME	total_index_build_time monitor element
TOTAL_INDEX_BUILD_PROC_TIME	total_index_build_proc_time monitor element
TOTAL_INDEXES_BUILT	total_indexes_built monitor element
TOTAL_PEAS	total_peas - Total partial early aggregations
TOTAL_PEDS	total_peds - Total partial early distincts
TOTAL_ROUTINE_INVOCATIONS	total_routine_invocations - Total routine invocations
TOTAL_ROUTINE_NON_SECT_PROC_TIME	total_routine_non_sect_proc_time - Non-section processing time
TOTAL_ROUTINE_NON_SECT_TIME	total_routine_non_sect_time - Non-section routine execution time
TOTAL_ROUTINE_TIME	total_routine_time - Total routine time
TOTAL_ROUTINE_USER_CODE_PROC_TIME	total_routine_user_code_proc_time - Total routine user code processing time
TOTAL_ROUTINE_USER_CODE_TIME	total_routine_user_code_time - Total routine user code time
TOTAL_SECTION_PROC_TIME	total_section_proc_time - Total section processing time
TOTAL_SECTION_SORT_PROC_TIME	total_section_sort_proc_time - Total section sort processing time

Table 100. Metric names returned by MON\_FORMAT\_XML\_METRICS\_BY\_ROW for XML documents containing an activity\_metrics element type (continued)

Metric Name	Description or Monitor element
TOTAL_SECTION_SORT_TIME	total_section_sort_time - Total section sort time
TOTAL_SECTION_SORTS	total_section_sorts - Total section sorts
TOTAL_SECTION_TIME	total_section_time - Total section time
TOTAL_SORTS	total_sorts - Total sorts
TQ_SORT_HEAP_REJECTIONS	tq_sort_heap_rejections - Table queue sort heap rejections
TQ_SORT_HEAP_REQUESTS	tq_sort_heap_requests - Table queue sort heap requests
TQ_TOT_SEND_SPILLS	tq_tot_send_spills - Total number of table queue buffers overflowed
WLM_QUEUE_ASSIGNMENTS_TOTAL	wlm_queue_assignments_total - Workload manager total queue assignments
WLM_QUEUE_TIME_TOTAL	wlm_queue_time_total - Workload manager total queue time

## Example

This example shows how to call the MON\_FORMAT\_XML\_METRICS\_BY\_ROW table function to return row-based formatted information from the XML document produced by the MON\_GET\_WORKLOAD\_DETAILS table function.

```
SELECT SUBSTR(TFXML.WORKLOAD_NAME, 1, 13) AS WORKLOAD_NAME,
 SUBSTR(METRICS.METRIC_NAME, 1, 25) AS METRIC_NAME,
 METRICS.VALUE
FROM
 TABLE(MON_GET_WORKLOAD_DETAILS(NULL, -2)) AS TFXML,
 TABLE(MON_FORMAT_XML_METRICS_BY_ROW(TFXML.DETAILS)) AS METRICS
ORDER BY METRICS.VALUE DESC
```

The following is a partial listing of the output of this query.

WORKLOAD_NAME	METRIC_NAME	VALUE
PAYROLL	ACT_COMPLETED_TOTAL	15
FINANCE	ACT_COMPLETED_TOTAL	12
PAYROLL	LOCK_WAITS	8
FINANCE	LOCK_WAITS	5
FINANCE	DEADLOCKS	3
PAYROLL	DEADLOCKS	0

## MON\_FORMAT\_XML\_TIMES\_BY\_ROW - Get formatted row-based combined hierarchy wait and processing times

The MON\_FORMAT\_XML\_TIMES\_BY\_ROW table function returns formatted row based output for the combined hierarchy of wait and processing times that are contained in an XML metrics document.

## Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

## Syntax

►►—MON\_FORMAT\_XML\_TIMES\_BY\_ROW—(—*xml doc*—)————►►

The schema is SYSPROC.

## Routine parameters

### *xml doc*

An input argument of type BLOB(8M) that contains an XML document with either a *system\_metrics* or *activity\_metrics* element. XML documents with these elements can be obtained from the following sources:

- Returned by one of the MON\_GET\_\*\_DETAILS table functions.
- From the metrics column output by statistics and activity event monitors.
- From the formatted output of the unit of work, or package cache event monitors.

## Information returned

Table 101. Information returned for MON\_FORMAT\_XML\_TIMES\_BY\_ROW

Column Name	Data Type	Description
METRIC_NAME	VARCHAR(128)	The unique identifier for the total time metric value.
TOTAL_TIME_VALUE	BIGINT	The total time value in milliseconds corresponding to metric_name.
COUNT	BIGINT	Number of occurrences of this type of interval.
PARENT_METRIC_NAME	VARCHAR(128)	The identifier of the parent total time metric whose value contains the total_time_value as a subset.

XML documents that contain an element of type *system\_metrics* are generated from the following interfaces:

- MON\_GET\_CONNECTION\_DETAILS
- MON\_GET\_SERVICE\_SUBCLASS\_DETAILS
- MON\_GET\_UNIT\_OF\_WORK\_DETAILS
- MON\_GET\_WORKLOAD\_DETAILS
- DETAILS\_XML column from a STATISTICS event monitor
- METRICS column produced by EVMON\_FORMAT\_UE\_TO\_TABLES for the UNIT OF WORK event monitor
- XMLREPORT column of EVMON\_FORMAT\_UE\_TO\_XML for the UNIT OF WORK event monitor

See Table 102 on page 506 for the types of metrics and their parent metrics that are returned from the XML document in this case:

Table 102. Metric names returned by MON\_FORMAT\_XML\_TIMES\_BY\_ROW for XML documents containing a system\_metrics element type

Metric Name	Parent Metric Name	Description of metric or Monitor element
AGENT_WAIT_TIME	TOTAL_WAIT_TIME	agent_wait_time - Agent wait time
AUDIT_FILE_WRITE_WAIT_TIME	TOTAL_WAIT_TIME	audit_file_write_wait_time - Audit file write wait time
AUDIT_SUBSYSTEM_WAIT_TIME	TOTAL_WAIT_TIME	audit_subsystem_wait_time - Audit subsystem wait time
CF_WAIT_TIME	TOTAL_WAIT_TIME	cf_wait_time - cluster caching facility wait time
CLIENT_IDLE_WAIT_TIME	NULL	client_idle_wait_time - Client idle wait time
COMM_EXIT_WAIT_TIME	TOTAL_WAIT_TIME	comm_exit_wait_time - Communication buffer exit wait time monitor element
DIAGLOG_WRITE_WAIT_TIME	TOTAL_WAIT_TIME	diaglog_write_wait_time - Diagnostic log file write wait time
DIRECT_READ_TIME	TOTAL_WAIT_TIME	direct_read_time - Direct read time
DIRECT_WRITE_TIME	TOTAL_WAIT_TIME	direct_write_time - Direct write time
EVMON_WAIT_TIME	TOTAL_WAIT_TIME	evmon_wait_time - Event monitor wait time
FCM_MESSAGE_RECV_WAIT_TIME	FCM_RECV_WAIT_TIME	fcm_message_recv_wait_time - FCM message received wait time
FCM_MESSAGE_SEND_WAIT_TIME	FCM_SEND_WAIT_TIME	fcm_message_send_wait_time - FCM message send wait time
FCM_RECV_WAIT_TIME	TOTAL_WAIT_TIME	fcm_recv_wait_time - FCM received wait time
FCM_SEND_WAIT_TIME	TOTAL_WAIT_TIME	fcm_send_wait_time - FCM send wait time
FCM_TQ_RECV_WAIT_TIME	FCM_RECV_WAIT_TIME	fcm_tq_recv_wait_time - FCM table queue received wait time
FCM_TQ_SEND_WAIT_TIME	FCM_SEND_WAIT_TIME	fcm_tq_send_wait_time - FCM table queue send wait time
IDA_RECV_WAIT_TIME	TOTAL_WAIT_TIME	The total amount of time spent waiting to receive data from an in-database analytics process.
IDA_SEND_WAIT_TIME	TOTAL_WAIT_TIME	The total amount of time spent waiting to send data to an in-database analytics process.
IPC_RECV_WAIT_TIME	TOTAL_WAIT_TIME	ipc_recv_wait_time - Interprocess communication received wait time
IPC_SEND_WAIT_TIME	TOTAL_WAIT_TIME	ipc_send_wait_time - Interprocess communication send wait time
LOCK_WAIT_TIME_GLOBAL	LOCK_WAIT_TIME	lock_wait_time_global - Lock wait time global
LOCK_WAIT_TIME	TOTAL_WAIT_TIME	lock_wait_time - Time waited on locks
LOG_BUFFER_WAIT_TIME	TOTAL_WAIT_TIME	log_buffer_wait_time - Log buffer wait time
LOG_DISK_WAIT_TIME	TOTAL_WAIT_TIME	log_disk_wait_time - Log disk wait time
POOL_READ_TIME	TOTAL_WAIT_TIME	pool_read_time - Total buffer pool physical read time
POOL_WRITE_TIME	TOTAL_WAIT_TIME	pool_write_time - Total buffer pool physical write time
PREFETCH_WAIT_TIME	TOTAL_WAIT_TIME	prefetch_wait_time - Time waited for prefetch
RECLAIM_WAIT_TIME	TOTAL_WAIT_TIME	reclaim_wait_time - Reclaim wait time
SPACEMAPPAGE_RECLAIM_WAIT_TIME	TOTAL_WAIT_TIME	spacemappage_reclaim_wait_time - Space map page reclaim wait time

Table 102. Metric names returned by MON\_FORMAT\_XML\_TIMES\_BY\_ROW for XML documents containing a system\_metrics element type (continued)

Metric Name	Parent Metric Name	Description of metric or Monitor element
TCPIP_RECV_WAIT_TIME	TOTAL_WAIT_TIME	tcipip_recv_wait_time - TCP/IP received wait time
TCPIP_SEND_WAIT_TIME	TOTAL_WAIT_TIME	tcipip_send_wait_time - TCP/IP send wait time
TOTAL_BACKUP_PROC_TIME	TOTAL_RQST_TIME	total_backup_proc_time monitor element
TOTAL_COL_PROC_TIME	TOTAL_SECTION_PROC_TIME	total_col_proc_time - Total column-oriented processing time
TOTAL_COL_SYNOPSIS_PROC_TIME	TOTAL_COL_PROC_TIME	total_col_synopsis_proc_time - Total column-organized synopsis table processing time
TOTAL_COMMIT_PROC_TIME	TOTAL_RQST_TIME	total_commit_proc_time - Total commits processing time
TOTAL_COMPILE_PROC_TIME	TOTAL_RQST_TIME	total_compile_proc_time - Total compile processing time
TOTAL_CONNECT_AUTHENTICATION_PROC_TIME	TOTAL_CONNECT_REQUEST_PROC_TIME	total_connect_authentication_proc_time - Total connection authentication processing time
TOTAL_CONNECT_REQUEST_PROC_TIME	TOTAL_RQST_TIME	total_connect_request_proc_time - Total connection or switch user request processing time
TOTAL_EXTENDED_LATCH_WAIT_TIME	TOTAL_WAIT_TIME	total_extended_latch_wait_time - Total extended latch wait time
TOTAL_IMPLICIT_COMPILE_PROC_TIME	TOTAL_RQST_TIME	total_implicit_compile_proc_time - Total implicit compile processing time
TOTAL_INDEX_BUILD_PROC_TIME	TOTAL_SECTION_PROC_TIME	total_index_build_proc_time monitor element
TOTAL_LOAD_PROC_TIME	TOTAL_RQST_TIME	total_load_proc_time - Total load processing time
TOTAL_REORG_PROC_TIME	TOTAL_RQST_TIME	total_reorg_proc_time - Total reorganization processing time
TOTAL_ROLLBACK_PROC_TIME	TOTAL_RQST_TIME	total_rollback_proc_time - Total rollback processing time
TOTAL_ROUTINE_USER_CODE_PROC_TIME	TOTAL_RQST_TIME	total_routine_user_code_proc_time - Total routine user code processing time
TOTAL_RQST_TIME	NULL	total_rqst_time - Total request time
TOTAL_RUNSTATS_PROC_TIME	TOTAL_RQST_TIME	total_runstats_proc_time - Total runtime statistics processing time
TOTAL_SECTION_PROC_TIME	TOTAL_RQST_TIME	total_section_proc_time - Total section processing time
TOTAL_SECTION_SORT_PROC_TIME	TOTAL_SECTION_PROC_TIME	total_section_sort_proc_time - Total section sort processing time
TOTAL_STATS_FABRICATION_PROC_TIME	TOTAL_COMPILE_PROC_TIME	total_stats_fabrication_proc_time - Total statistics fabrication processing time
TOTAL_SYNC_RUNSTATS_PROC_TIME	TOTAL_COMPILE_PROC_TIME	total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time
TOTAL_WAIT_TIME	TOTAL_RQST_TIME	total_wait_time - Total wait time
WLM_QUEUE_TIME_TOTAL	TOTAL_WAIT_TIME	wlm_queue_time_total - Workload manager total queue time

XML documents that contain an element of type *activity\_metrics* are generated from the following interfaces:

- MON\_GET\_ACTIVITY\_DETAILS
- MON\_GET\_PKG\_CACHE\_STMT\_DETAILS
- DETAILS\_XML column from an ACTIVITY event monitor
- METRICS column produced by EVMON\_FORMAT\_UE\_TO\_TABLES for the PACKAGE CACHE event monitor
- XMLREPORT column of EVMON\_FORMAT\_UE\_TO\_XML for the PACKAGE CACHE event monitor

See Table 103 for the types of metrics and their parent metrics that are returned from the XML document in this case:

*Table 103. Metric names returned by MON\_FORMAT\_XML\_TIMES\_BY\_ROW for XML documents containing an activity\_metrics element type*

Metric Name	Parent Metric Name	Description or Monitor element
AUDIT_FILE_WRITE_WAIT_TIME	TOTAL_ACT_WAIT_TIME	audit_file_write_wait_time - Audit file write wait time
AUDIT_SUBSYSTEM_WAIT_TIME	TOTAL_ACT_WAIT_TIME	audit_subsystem_wait_time - Audit subsystem wait time
CF_WAIT_TIME	TOTAL_ACT_WAIT_TIME	cf_wait_time - cluster caching facility wait time
DIAGLOG_WRITE_WAIT_TIME	TOTAL_ACT_WAIT_TIME	diaglog_write_wait_time - Diagnostic log file write wait time
DIRECT_READ_TIME	TOTAL_ACT_WAIT_TIME	direct_read_time - Direct read time
DIRECT_WRITE_TIME	TOTAL_ACT_WAIT_TIME	direct_write_time - Direct write time
EVMON_WAIT_TIME	TOTAL_ACT_WAIT_TIME	evmon_wait_time - Event monitor wait time
FCM_MESSAGE_RECV_WAIT_TIME	FCM_RECV_WAIT_TIME	fcm_message_recv_wait_time - FCM message received wait time
FCM_MESSAGE_SEND_WAIT_TIME	FCM_SEND_WAIT_TIME	fcm_message_send_wait_time - FCM message send wait time
FCM_RECV_WAIT_TIME	TOTAL_ACT_WAIT_TIME	fcm_recv_wait_time - FCM received wait time
FCM_SEND_WAIT_TIME	TOTAL_ACT_WAIT_TIME	fcm_send_wait_time - FCM send wait time
FCM_TQ_RECV_WAIT_TIME	FCM_RECV_WAIT_TIME	fcm_tq_recv_wait_time - FCM table queue received wait time
FCM_TQ_SEND_WAIT_TIME	FCM_SEND_WAIT_TIME	fcm_tq_send_wait_time - FCM table queue send wait time
IDA_RECV_WAIT_TIME	TOTAL_WAIT_TIME	The total amount of time spent waiting to receive data from an in-database analytics process.
IDA_SEND_WAIT_TIME	TOTAL_WAIT_TIME	The total amount of time spent waiting to send data to an in-database analytics process.
LOCK_WAIT_TIME_GLOBAL	LOCK_WAIT_TIME	lock_wait_time_global - Lock wait time global
LOCK_WAIT_TIME	TOTAL_ACT_WAIT_TIME	lock_wait_time - Time waited on locks
LOG_BUFFER_WAIT_TIME	TOTAL_ACT_WAIT_TIME	log_buffer_wait_time - Log buffer wait time
LOG_DISK_WAIT_TIME	TOTAL_ACT_WAIT_TIME	log_disk_wait_time - Log disk wait time
POOL_READ_TIME	TOTAL_ACT_WAIT_TIME	pool_read_time - Total buffer pool physical read time
POOL_WRITE_TIME	TOTAL_ACT_WAIT_TIME	pool_write_time - Total buffer pool physical write time

Table 103. Metric names returned by MON\_FORMAT\_XML\_TIMES\_BY\_ROW for XML documents containing an activity\_metrics element type (continued)

Metric Name	Parent Metric Name	Description or Monitor element
PREFETCH_WAIT_TIME	TOTAL_ACT_WAIT_TIME	prefetch_wait_time - Time waited for prefetch
RECLAIM_WAIT_TIME	TOTAL_ACT_WAIT_TIME	reclaim_wait_time - Reclaim wait time
SPACEMAPPAGE_RECLAIM_WAIT_TIME	TOTAL_ACT_WAIT_TIME	spacemappage_reclaim_wait_time - Space map page reclaim wait time
STMT_EXEC_TIME	NULL	stmt_exec_time - Statement execution time
TOTAL_ACT_WAIT_TIME	STMT_EXEC_TIME	total_act_wait_time - Total activity wait time
TOTAL_COL_PROC_TIME	TOTAL_SECTION_PROC_TIME	total_col_proc_time - Total column-oriented processing time
TOTAL_EXTENDED_LATCH_WAIT_TIME	TOTAL_ACT_WAIT_TIME	total_extended_latch_wait_time - Total extended latch wait time
TOTAL_INDEX_BUILD_PROC_TIME	TOTAL_SECTION_PROC_TIME	total_index_build_proc_time monitor element
TOTAL_ROUTINE_NON_SECT_PROC_TIME	STMT_EXEC_TIME	total_routine_non_sect_proc_time - Non-section processing time
TOTAL_ROUTINE_USER_CODE_PROC_TIME	TOTAL_ROUTINE_NON_SECT_PROC_TIME	total_routine_user_code_proc_time - Total routine user code processing time
TOTAL_SECTION_PROC_TIME	STMT_EXEC_TIME	total_section_proc_time - Total section processing time
TOTAL_SECTION_SORT_PROC_TIME	TOTAL_SECTION_PROC_TIME	total_section_sort_proc_time - Total section sort processing time
WLM_QUEUE_TIME_TOTAL	NULL	wlm_queue_time_total - Workload manager total queue time

## Example

To determine where time is being spent by your application within the database manager, you can run the following query to show the combined wait and processing time metrics in the metrics hierarchy.

```
SELECT SUBSTR(T.SERVICE_SUPERCLASS_NAME,1,15) as SUPERCLASS,
 SUBSTR(T.SERVICE_SUBCLASS_NAME,1,15) as SUBCLASS,
 T.MEMBER,
 SUBSTR(U.METRIC_NAME, 1,15) AS METRIC_NAME,
 SUBSTR(U.PARENT_METRIC_NAME,1,15) AS PARENT_NAME,
 U.TOTAL_TIME_VALUE,
 U.COUNT
FROM
TABLE(MON_GET_SERVICE_SUBCLASS_DETAILS(NULL, NULL, -2)) AS T,
TABLE(MON_FORMAT_XML_TIMES_BY_ROW(T.DETAILS)) AS U
```

The following is an example of output from this query.

SUPERCLASS	SUBCLASS	MEMBER	METRIC_NAME	PARENT_NAME	T..._VALUE	COUNT
MYSC	MYSSC	0	FCM_MESSAGE_REC	FCM_RECV_WAIT_T	0	0
MYSC	MYSSC	0	FCM_TQ_RECV_WAI	FCM_RECV_WAIT_T	0	0
MYSC	MYSSC	0	FCM_MESSAGE_SEN	FCM_SEND_WAIT_T	0	0
MYSC	MYSSC	0	FCM_TQ_SEND_WAI	FCM_SEND_WAIT_T	0	0
MYSC	MYSSC	0	TOTAL_COMMIT_PR	TOTAL_RQST_TIME	300	1
MYSC	MYSSC	0	TOTAL_COMPILE_P	TOTAL_RQST_TIME	700	1
MYSC	MYSSC	0	TOTAL_IMPLICIT	TOTAL_RQST_TIME	0	0
MYSC	MYSSC	0	TOTAL_LOAD_PROC	TOTAL_RQST_TIME	0	0
MYSC	MYSSC	0	TOTAL_REORG_PRO	TOTAL_RQST_TIME	0	0

MYSC	MYSSC	0	TOTAL_ROLLBACK_	TOTAL_RQST_TIME	0	0
MYSC	MYSSC	0	TOTAL_RUNSTATS_	TOTAL_RQST_TIME	0	0
MYSC	MYSSC	0	TOTAL_SECTION_P	TOTAL_RQST_TIME	7322	1
MYSC	MYSSC	0	TOTAL_WAIT_TIME	TOTAL_RQST_TIME	0	0
MYSC	MYSSC	0	TOTAL_SECTION_S	TOTAL_SECTION_P	0	0
MYSC	MYSSC	0	AGENT_WAIT_TIME	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	AUDIT_FILE_WRIT	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	AUDIT_SUBSYSTEM	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	DIAGLOG_WRITE_W	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	DIRECT_READ_TIM	TOTAL_WAIT_TIME	1204	17
MYSC	MYSSC	0	DIRECT_WRITE_TI	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	FCM_RECV_WAIT_T	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	FCM_SEND_WAIT_T	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	IPC_RECV_WAIT_T	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	IPC_SEND_WAIT_T	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	LOCK_WAIT_TIME	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	LOG_BUFFER_WAIT	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	LOG_DISK_WAIT_T	TOTAL_WAIT_TIME	523	2
MYSC	MYSSC	0	POOL_READ_TIME	TOTAL_WAIT_TIME	2432	7
MYSC	MYSSC	0	POOL_WRITE_TIME	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	TCPIP_RECV_WAIT	TOTAL_WAIT_TIME	523	1
MYSC	MYSSC	0	TCPIP_SEND_WAIT	TOTAL_WAIT_TIME	241	1
MYSC	MYSSC	0	WLM_QUEUE_TIME	TOTAL_WAIT_TIME	0	0
MYSC	MYSSC	0	CLIENT_IDLE_WAI	-	234	-
MYSC	MYSSC	0	TOTAL_RQST_TIME	-	13245	1

34 record(s) selected.

## MON\_FORMAT\_XML\_WAIT\_TIMES\_BY\_ROW - Get formatted row-based output for wait times

The MON\_FORMAT\_XML\_WAIT\_TIMES\_BY\_ROW table function returns formatted row-based output for the wait times contained in an XML metrics document.

### Syntax

►►—MON\_FORMAT\_XML\_WAIT\_TIMES\_BY\_ROW—(—*xml doc*—)—————◄◄

The schema is SYSPROC.

### Table function parameters

#### *xml doc*

An input argument of type BLOB(8M) that contains an XML document with either a system\_metrics or activity\_metrics element. XML documents with these elements can be obtained from the following sources:

- Returned by one of the MON\_GET\_\*\_DETAILS table functions.
- From the metrics column output by statistics and activity event monitors.
- From the formatted output of the unit of work, or package cache event monitors.

### Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

None

### Example

This example shows how to call the `MON_FORMAT_XML_WAIT_TIMES_BY_ROW` table function to return formatted row-based output from the XML document produced by the `MON_GET_WORKLOAD_DETAILS` table function. The output shows the metrics and their values for each workload.

```
SELECT SUBSTR(TFXML.WORKLOAD_NAME, 1, 13) AS WORKLOAD_NAME,
 SUBSTR(WAITS.METRIC_NAME, 1, 25) AS METRIC_NAME,
 WAITS.TOTAL_TIME_VALUE,
 WAITS.COUNT
FROM
 TABLE(MON_GET_WORKLOAD_DETAILS(NULL, -2)) AS TFXML,
 TABLE(MON_FORMAT_XML_WAIT_TIMES_BY_ROW(
 TFXML.DETAILS
)) AS WAITS
ORDER BY WAITS.TOTAL_TIME_VALUE DESC
```

The following is a partial listing of the output of this query.

WORKLOAD_NAME	METRIC_NAME	TOTAL_TIME_VALUE	COUNT
PAYROLL	CLIENT_IDLE_WAIT_TIME	2193672	174
FINANCE	CLIENT_IDLE_WAIT_TIME	738290	16
PAYROLL	DIRECT_READ_TIME	67892	81
FINANCE	DIRECT_READ_TIME	32343	8
FINANCE	LOCK_WAIT_TIME	8463	3
PAYROLL	LOCK_WAIT_TIME	55	1

### Information returned

Table 104. Information returned for `MON_FORMAT_XML_WAIT_TIMES_BY_ROW`

Column Name	Data Type	Description
METRIC_NAME	VARCHAR(128)	The unique identifier for the total time metric value.
TOTAL_TIME_VALUE	BIGINT	The total time value in milliseconds corresponding to metric_name.
COUNT	BIGINT	Number of occurrences of this type of interval.
PARENT_METRIC_NAME	VARCHAR(128)	The identifier of the parent total time metric whose value contains the total_time_value as a subset.

XML documents that contain an element of type *system\_metrics* are generated from the following interfaces:

- `MON_GET_CONNECTION_DETAILS`
- `MON_GET_SERVICE_SUBCLASS_DETAILS`
- `MON_GET_UNIT_OF_WORK_DETAILS`
- `MON_GET_WORKLOAD_DETAILS`
- `DETAILS_XML` column from a `STATISTICS` event monitor
- `METRICS` column produced by `EVMON_FORMAT_UE_TO_TABLES` for the `UNIT OF WORK` event monitor
- `XMLREPORT` column of `EVMON_FORMAT_UE_TO_XML` for the `UNIT OF WORK` event monitor

See Table 105 for the types of metrics and their parent metrics that are returned from the XML document in this case:

*Table 105. Metric names returned by MON\_FORMAT\_XML\_WAIT\_TIMES\_BY\_ROW for XML documents containing a system\_metrics element type*

Metric Name	Parent Metric Name	Description of metric or Monitor element
WLM_QUEUE_TIME_TOTAL	TOTAL_WAIT_TIME	wlm_queue_time_total - Workload manager total queue time
FCM_TQ_RECV_WAIT_TIME	FCM_RECV_WAIT_TIME	fcm_tq_recv_wait_time - FCM table queue received wait time
FCM_MESSAGE_RECV_WAIT_TIME	FCM_RECV_WAIT_TIME	fcm_message_recv_wait_time - FCM message received wait time
FCM_TQ_SEND_WAIT_TIME	FCM_SEND_WAIT_TIME	fcm_tq_send_wait_time - FCM table queue send wait time
FCM_MESSAGE_SEND_WAIT_TIME	FCM_SEND_WAIT_TIME	fcm_message_send_wait_time - FCM message send wait time
AGENT_WAIT_TIME	TOTAL_WAIT_TIME	agent_wait_time - Agent wait time
LOCK_WAIT_TIME	TOTAL_WAIT_TIME	lock_wait_time - Time waited on locks
DIRECT_READ_TIME	TOTAL_WAIT_TIME	direct_read_time - Direct read time
DIRECT_WRITE_TIME	TOTAL_WAIT_TIME	direct_write_time - Direct write time
LOG_BUFFER_WAIT_TIME	TOTAL_WAIT_TIME	log_buffer_wait_time - Log buffer wait time
LOG_DISK_WAIT_TIME	TOTAL_WAIT_TIME	log_disk_wait_time - Log disk wait time
TCPIP_RECV_WAIT_TIME	TOTAL_WAIT_TIME	tcpip_recv_wait_time - TCP/IP received wait time
CLIENT_IDLE_WAIT_TIME	NULL	client_idle_wait_time - Client idle wait time
IPC_RECV_WAIT_TIME	TOTAL_WAIT_TIME	ipc_recv_wait_time - Interprocess communication received wait time
IPC_SEND_WAIT_TIME	TOTAL_WAIT_TIME	ipc_send_wait_time - Interprocess communication send wait time
TCPIP_SEND_WAIT_TIME	TOTAL_WAIT_TIME	tcpip_send_wait_time - TCP/IP send wait time
POOL_WRITE_TIME	TOTAL_WAIT_TIME	pool_write_time - Total buffer pool physical write time
POOL_READ_TIME	TOTAL_WAIT_TIME	pool_read_time - Total buffer pool physical read time
AUDIT_FILE_WRITE_WAIT_TIME	TOTAL_WAIT_TIME	audit_file_write_wait_time - Audit file write wait time
AUDIT_SUBSYSTEM_WAIT_TIME	TOTAL_WAIT_TIME	audit_subsystem_wait_time - Audit subsystem wait time
DIAGLOG_WRITE_WAIT_TIME	TOTAL_WAIT_TIME	diaglog_write_wait_time - Diagnostic log file write wait time
FCM_SEND_WAIT_TIME	TOTAL_WAIT_TIME	fcm_send_wait_time - FCM send wait time
FCM_RECV_WAIT_TIME	TOTAL_WAIT_TIME	fcm_recv_wait_time - FCM received wait time
TOTAL_WAIT_TIME	TOTAL_RQST_TIME	total_wait_time - Total wait time
LOCK_WAIT_TIME_GLOBAL	LOCK_WAIT_TIME	lock_wait_time_global - Lock wait time global
RECLAIM_WAIT_TIME	TOTAL_WAIT_TIME	reclaim_wait_time - Reclaim wait time
SPACEMAPPAGE_RECLAIM_WAIT_TIME	TOTAL_WAIT_TIME	spacemappage_reclaim_wait_time - Space map page reclaim wait time
CF_WAIT_TIME	TOTAL_WAIT_TIME	cf_wait_time - cluster caching facility wait time
EVMON_WAIT_TIME	TOTAL_WAIT_TIME	evmon_wait_time - Event monitor wait time

Table 105. Metric names returned by MON\_FORMAT\_XML\_WAIT\_TIMES\_BY\_ROW for XML documents containing a system\_metrics element type (continued)

Metric Name	Parent Metric Name	Description of metric or Monitor element
TOTAL_EXTENDED_LATCH_WAIT_TIME	TOTAL_WAIT_TIME	total_extended_latch_wait_time - Total extended latch wait time
PREFETCH_WAIT_TIME	TOTAL_WAIT_TIME	prefetch_wait_time - Time waited for prefetch
COMM_EXIT_WAIT_TIME	TOTAL_WAIT_TIME	comm_exit_wait_time - Communication buffer exit wait time monitor element
IDA_SEND_WAIT_TIME	TOTAL_WAIT_TIME	The total amount of time spent waiting to send data to an in-database analytics process.
IDA_RECV_WAIT_TIME	TOTAL_WAIT_TIME	The total amount of time spent waiting to receive data from an in-database analytics process.

XML documents that contain an element of type *activity\_metrics* are generated from the following interfaces:

- MON\_GET\_ACTIVITY\_DETAILS
- MON\_GET\_PKG\_CACHE\_STMT\_DETAILS
- DETAILS\_XML column from an ACTIVITY event monitor
- METRICS column produced by EVMON\_FORMAT\_UE\_TO\_TABLES for the PACKAGE CACHE event monitor
- XMLREPORT column of EVMON\_FORMAT\_UE\_TO\_XML for the PACKAGE CACHE event monitor

See Table 106 for the types of metrics and their parent metrics that are returned from the XML document in this case:

Table 106. Metric names returned by MON\_FORMAT\_XML\_WAIT\_TIMES\_BY\_ROW for XML documents containing an activity\_metrics element type

Metric Name	Parent Metric Name	Description or Monitor element
FCM_TQ_RECV_WAIT_TIME	FCM_RECV_WAIT_TIME	fcm_tq_recv_wait_time - FCM table queue received wait time
FCM_MESSAGE_RECV_WAIT_TIME	FCM_RECV_WAIT_TIME	fcm_message_recv_wait_time - FCM message received wait time
FCM_TQ_SEND_WAIT_TIME	FCM_SEND_WAIT_TIME	fcm_tq_send_wait_time - FCM table queue send wait time
FCM_MESSAGE_SEND_WAIT_TIME	FCM_SEND_WAIT_TIME	fcm_message_send_wait_time - FCM message send wait time
LOCK_WAIT_TIME	TOTAL_ACT_WAIT_TIME	lock_wait_time - Time waited on locks
DIRECT_READ_TIME	TOTAL_ACT_WAIT_TIME	direct_read_time - Direct read time
DIRECT_WRITE_TIME	TOTAL_ACT_WAIT_TIME	direct_write_time - Direct write time
LOG_BUFFER_WAIT_TIME	TOTAL_ACT_WAIT_TIME	log_buffer_wait_time - Log buffer wait time
LOG_DISK_WAIT_TIME	TOTAL_ACT_WAIT_TIME	log_disk_wait_time - Log disk wait time
POOL_WRITE_TIME	TOTAL_ACT_WAIT_TIME	pool_write_time - Total buffer pool physical write time
POOL_READ_TIME	TOTAL_ACT_WAIT_TIME	pool_read_time - Total buffer pool physical read time
AUDIT_FILE_WRITE_WAIT_TIME	TOTAL_ACT_WAIT_TIME	audit_file_write_wait_time - Audit file write wait time

Table 106. Metric names returned by MON\_FORMAT\_XML\_WAIT\_TIMES\_BY\_ROW for XML documents containing an activity\_metrics element type (continued)

Metric Name	Parent Metric Name	Description or Monitor element
AUDIT_SUBSYSTEM_WAIT_TIME	TOTAL_ACT_WAIT_TIME	audit_subsystem_wait_time - Audit subsystem wait time
DIAGLOG_WRITE_WAIT_TIME	TOTAL_ACT_WAIT_TIME	diaglog_write_wait_time - Diagnostic log file write wait time
FCM_SEND_WAIT_TIME	TOTAL_ACT_WAIT_TIME	fcm_send_wait_time - FCM send wait time
FCM_RECV_WAIT_TIME	TOTAL_ACT_WAIT_TIME	fcm_recv_wait_time - FCM received wait time
TOTAL_ACT_WAIT_TIME	STMT_EXEC_TIME	total_act_wait_time - Total activity wait time
LOCK_WAIT_TIME_GLOBAL	LOCK_WAIT_TIME	lock_wait_time_global - Lock wait time global
RECLAIM_WAIT_TIME	TOTAL_ACT_WAIT_TIME	reclaim_wait_time - Reclaim wait time
SPACEMAPPAGE_RECLAIM_WAIT_TIME	TOTAL_ACT_WAIT_TIME	spacemappage_reclaim_wait_time - Space map page reclaim wait time
CF_WAIT_TIME	TOTAL_ACT_WAIT_TIME	cf_wait_time - cluster caching facility wait time
EVMON_WAIT_TIME	TOTAL_ACT_WAIT_TIME	evmon_wait_time - Event monitor wait time
TOTAL_EXTENDED_LATCH_WAIT_TIME	TOTAL_ACT_WAIT_TIME	total_extended_latch_wait_time - Total extended latch wait time
PREFETCH_WAIT_TIME	TOTAL_ACT_WAIT_TIME	prefetch_wait_time - Time waited for prefetch
WLM_QUEUE_TIME_TOTAL	NULL	wlm_queue_time_total - Workload manager total queue time
IDA_SEND_WAIT_TIME	TOTAL_WAIT_TIME	The total amount of time spent waiting to send data to an in-database analytics process.
IDA_RECV_WAIT_TIME	TOTAL_WAIT_TIME	The total amount of time spent waiting to receive data from an in-database analytics process.

### MON\_GET\_ACTIVITY table function - Return a list of activities

The MON\_GET\_ACTIVITY function returns the list of all activities that were submitted by the specified application on the specified member and have not yet been completed.

Refer to Table 107 on page 515 for a complete list of information that can be returned.

### Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

### Default PUBLIC privilege

None

## Syntax

►►—MON\_GET\_ACTIVITY—(—*application\_handle*—,—*member*—)—►►

The schema is SYSPROC.

## Routine parameters

### *application\_handle*

An input argument of type BIGINT that specifies an application handle for which a list of activities is to be returned. If the argument is null, the data is retrieved for all the applications in the database.

### *member*

An input argument of type INTEGER that specifies a valid member number in the same instance as the currently connected database. Specify -1 for the current member, or -2 for all active members. If the null value is specified, -1 is set implicitly.

## Information returned

Table 107. Information returned by the MON\_GET\_ACTIVITY routine

Column name	Data type	Description
APPLICATION_HANDLE	BIGINT	application_handle - Application handle monitor element
DBPARTITIONNUM	SMALLINT	dbpartitionnum - Database partition number monitor element
COORD_PARTITION_NUM	SMALLINT	coord_partition_num - Coordinator partition number monitor element
LOCAL_START_TIME	TIMESTAMP	local_start_time - Local start time monitor element
UOW_ID	INTEGER	uow_id - Unit of work ID monitor element
ACTIVITY_ID	INTEGER	activity_id - Activity ID monitor element
PARENT_UOW_ID	INTEGER	parent_uow_id - Parent unit of work ID monitor element
PARENT_ACTIVITY_ID	INTEGER	parent_activity_id - Parent activity ID monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
ACTIVITY_STATE	VARCHAR(32)	<p>activity_state - Activity state monitor element</p> <p>Possible values are as follows:</p> <p><b>QUEUED</b></p> <p>The activity is queued by a workload management queuing threshold. In a partitioned database environment, this state might mean that the coordinator agent has made an RPC to the catalog member to obtain threshold tickets and has not yet received a response. This state might indicate that the activity has been queued by a workload management queuing threshold or, if not much time has elapsed, can indicate that the activity is in the process of obtaining its tickets. To obtain a more accurate picture of whether the activity is being queued, determine what agent is working on the activity, and find out whether the EVENT_OBJECT value of the object at the catalog member has a value of WLM_QUEUE.</p> <p><b>TERMINATING</b></p> <p>The activity has finished running and is being removed from the system.</p>
ACTIVITY_TYPE	VARCHAR(32)	<p>activity_type - Activity type monitor element</p> <p>Possible values are as follows:</p> <ul style="list-style-type: none"> <li>• CALL</li> <li>• DDL</li> <li>• LOAD</li> <li>• OTHER</li> <li>• READ_DML</li> <li>• WRITE_DML</li> </ul>
NESTING_LEVEL	INTEGER	nesting_level - Nesting level monitor element
INVOCATION_ID	INTEGER	invocation_id - Invocation ID monitor element
ROUTINE_ID	INTEGER	routine_id - Routine ID monitor element
UTILITY_ID	INTEGER	utility_id - Utility ID monitor element
SERVICE_CLASS_ID	INTEGER	service_class_id - Service class ID monitor element
DATABASE_WORK_ACTION_SET_ID	INTEGER	<p>db_work_action_set_id - Database work action set ID monitor element</p> <p>One of the following values:</p> <ul style="list-style-type: none"> <li>• If this activity has been categorized into a work class of database scope, the value is the ID of the work class set of which this work class is a member.</li> <li>• If this activity has not been categorized into a work class of database scope, the value is null.</li> </ul>

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
DATABASE_WORK_CLASS_ID	INTEGER	db_work_class_id - Database work class ID monitor element  One of the following values: <ul style="list-style-type: none"> <li>If this activity has been categorized into a work class of database scope, the value is the ID of the work class.</li> <li>If this activity has not been categorized into a work class of database scope, the value is null.</li> </ul>
SERVICE_CLASS_WORK_ACTION_SET_ID	INTEGER	service_class_work_action_set_id - Service class work action set ID monitor element  One of the following values: <ul style="list-style-type: none"> <li>If this activity has been categorized into a work class of service class scope, the value is the ID of the work action set associated with the work class set to which the work class belongs.</li> <li>If this activity has not been categorized into a work class of service class scope, the value is null.</li> </ul>
SERVICE_CLASS_WORK_CLASS_ID	INTEGER	service_class_work_class_id - Service class work class ID monitor element  One of the following values: <ul style="list-style-type: none"> <li>If this activity has been categorized into a work class of service class scope, the value is the ID of the work class assigned to this activity.</li> <li>If this activity has not been categorized into a work class of service class scope, the value is null.</li> </ul>
EXECUTABLE_ID	VARCHAR(32) FOR BIT DATA	executable_id - Executable ID monitor element
TOTAL_CPU_TIME	BIGINT	total_cpu_time - Total CPU time
ROWS_READ	BIGINT	rows_read - Rows read
ROWS_RETURNED	BIGINT	rows_returned - Rows returned
QUERY_COST_ESTIMATE	BIGINT	query_cost_estimate - Query cost estimate
DIRECT_READS	BIGINT	direct_reads - Direct reads from database
DIRECT_WRITES	BIGINT	direct_writes - Direct writes to database
ENTRY_TIME	TIMESTAMP	entry_time - Entry time
MEMBER	SMALLINT	member - Database member monitor element
COORD_MEMBER	SMALLINT	coord_member - Coordinator member monitor element
PACKAGE_SCHEMA	VARCHAR(128)	package_schema - Package schema
PACKAGE_NAME	VARCHAR(128)	package_name - Package name
PACKAGE_VERSION_ID	VARCHAR(64)	package_version_id - Package version
SECTION_NUMBER	BIGINT	section_number - Section number
STMTNO	INTEGER	stmtno - Statement number monitor element
APPL_ID	VARCHAR(128)	appl_id - Application ID monitor element
CLIENT_ACCTNG	VARCHAR(255)	client_acctng - Client accounting string monitor element
CLIENT_APPLNAME	VARCHAR(255)	client_applname - Client application name monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
CLIENT_USERID	VARCHAR(255)	client_userid - Client user ID monitor element
CLIENT_WRKSTNNAME	VARCHAR(255)	client_wrkstnname - Client workstation name monitor element
EFFECTIVE_ISOLATION	CHAR(2)	effective_isolation - Effective isolation monitor element
EFFECTIVE_LOCK_TIMEOUT	BIGINT	effective_lock_timeout - Effective lock timeout monitor element
EFFECTIVE_QUERY_DEGREE	BIGINT	effective_query_degree - Effective query degree monitor element
ACTIVITYTOTALTIME_THRESHOLD_ID	INTEGER	activitytotaltime_threshold_id - Activity total time threshold ID monitor element
ACTIVITYTOTALTIME_THRESHOLD_VALUE	BIGINT	activitytotaltime_threshold_value - Activity total time threshold value monitor element
ACTIVITYTOTALTIME_THRESHOLD_VIOLATED	SMALLINT	activitytotaltime_threshold_violated - Activity total time threshold violated monitor element
AGGSQLTEMPSPACE_THRESHOLD_ID	INTEGER	aggsqltempespace_threshold_id - Aggregate SQL temporary space threshold ID monitor element
AGGSQLTEMPSPACE_THRESHOLD_VALUE	BIGINT	aggsqltempespace_threshold_value - AggSQL temporary space threshold value monitor element
AGGSQLTEMPSPACE_THRESHOLD_VIOLATED	SMALLINT	aggsqltempespace_threshold_violated - AggSQL temporary space threshold violated monitor element
CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_ID	INTEGER	concurrentdbcoordactivities_db_threshold_id - Concurrent database coordinator activities database threshold ID monitor element
CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_QUEUED	SMALLINT	concurrentdbcoordactivities_db_threshold_queued - Concurrent database coordinator activities database threshold queued monitor element
CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_VALUE	BIGINT	concurrentdbcoordactivities_db_threshold_value - Concurrent database coordinator activities database threshold value monitor element
CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_VIOLATED	SMALLINT	concurrentdbcoordactivities_db_threshold_violated - Concurrent database coordinator activities database threshold violated monitor element
CONCURRENTDBCOORDACTIVITIES_SUBCLASS_THRESHOLD_ID	INTEGER	concurrentdbcoordactivities_subclass_threshold_id - Concurrent database coordinator activities service subclass threshold ID monitor element
CONCURRENTDBCOORDACTIVITIES_SUBCLASS_THRESHOLD_QUEUED	SMALLINT	concurrentdbcoordactivities_subclass_threshold_queued - Concurrent database coordinator activities service subclass threshold queued monitor element
CONCURRENTDBCOORDACTIVITIES_SUBCLASS_THRESHOLD_VALUE	BIGINT	concurrentdbcoordactivities_subclass_threshold_value - Concurrent database coordinator activities service subclass threshold value monitor element
CONCURRENTDBCOORDACTIVITIES_SUBCLASS_THRESHOLD_VIOLATED	SMALLINT	concurrentdbcoordactivities_subclass_threshold_violated - Concurrent database coordinator activities service subclass threshold violated monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
CONCURRENTDBCOORDACTIVITIES_SUPERCLASS_THRESHOLD_ID	INTEGER	concurrentdbcoordactivities_superclass_threshold_id - Concurrent database coordinator activities service superclass threshold ID monitor element
CONCURRENTDBCOORDACTIVITIES_SUPERCLASS_THRESHOLD_QUEUED	SMALLINT	concurrentdbcoordactivities_superclass_threshold_queued - Concurrent database coordinator activities service superclass threshold queued monitor element
CONCURRENTDBCOORDACTIVITIES_SUPERCLASS_THRESHOLD_VALUE	BIGINT	concurrentdbcoordactivities_superclass_threshold_value - Concurrent database coordinator activities service superclass threshold value monitor element
CONCURRENTDBCOORDACTIVITIES_SUPERCLASS_THRESHOLD_VIOLATED	SMALLINT	concurrentdbcoordactivities_superclass_threshold_violated - Concurrent database coordinator activities service superclass threshold violated monitor element
CONCURRENTDBCOORDACTIVITIES_WL_WAS_THRESHOLD_ID	INTEGER	concurrentdbcoordactivities_wl_was_threshold_id - Concurrent database coordinator activities workload work action set threshold ID monitor element
CONCURRENTDBCOORDACTIVITIES_WL_WAS_THRESHOLD_QUEUED	SMALLINT	concurrentdbcoordactivities_wl_was_threshold_queued - Concurrent database coordinator activities workload work action set threshold queued monitor element
CONCURRENTDBCOORDACTIVITIES_WL_WAS_THRESHOLD_VALUE	BIGINT	concurrentdbcoordactivities_wl_was_threshold_value - Concurrent database coordinator activities workload work action set threshold value monitor element
CONCURRENTDBCOORDACTIVITIES_WL_WAS_THRESHOLD_VIOLATED	SMALLINT	concurrentdbcoordactivities_wl_was_threshold_violated - Concurrent database coordinator activities workload work action set threshold violated monitor element
CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET_THRESHOLD_ID	INTEGER	concurrentdbcoordactivities_work_action_set_threshold_id - Concurrent database coordinator activities work action set threshold ID monitor element
CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET_THRESHOLD_QUEUED	SMALLINT	concurrentdbcoordactivities_work_action_set_threshold_queued - Concurrent database coordinator activities work action set threshold queued monitor element
CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET_THRESHOLD_VALUE	BIGINT	concurrentdbcoordactivities_work_action_set_threshold_value - Concurrent database coordinator activities work action set threshold value monitor element
CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET_THRESHOLD_VIOLATED	SMALLINT	concurrentdbcoordactivities_work_action_set_threshold_violated - Concurrent database coordinator activities work action set threshold violated monitor element
CONCURRENTWORKLOADACTIVITIES_THRESHOLD_ID	INTEGER	concurrentworkloadactivities_threshold_id - Concurrent workload activities threshold ID monitor element
CONCURRENTWORKLOADACTIVITIES_THRESHOLD_VALUE	BIGINT	concurrentworkloadactivities_threshold_value - Concurrent workload activities threshold value monitor element
CONCURRENTWORKLOADACTIVITIES_THRESHOLD_VIOLATED	SMALLINT	concurrentworkloadactivities_threshold_violated - Concurrent workload activities threshold violated monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
CPUTIME_THRESHOLD_ID	INTEGER	cputime_threshold_id - CPU time threshold ID monitor element
CPUTIME_THRESHOLD_VALUE	BIGINT	cputime_threshold_value - CPU time threshold value monitor element
CPUTIME_THRESHOLD_VIOLATED	SMALLINT	cputime_threshold_violated - CPU time threshold violated monitor element
CPUTIMEINSC_THRESHOLD_ID	INTEGER	cputimeinsc_threshold_id - CPU time in service class threshold ID monitor element
CPUTIMEINSC_THRESHOLD_VALUE	BIGINT	cputimeinsc_threshold_value - CPU time in service class threshold value monitor element
CPUTIMEINSC_THRESHOLD_VIOLATED	SMALLINT	cputimeinsc_threshold_violated - CPU time in service class threshold violated monitor element
DATATAGINSC_THRESHOLD_ID	INTEGER	datataginsc_threshold_id - Data tag in service class threshold (IN condition) ID monitor element
DATATAGINSC_THRESHOLD_VALUE	VARCHAR(32)	datataginsc_threshold_value - Data tag in service class threshold (IN condition) value monitor element
DATATAGINSC_THRESHOLD_VIOLATED	SMALLINT	datataginsc_threshold_violated - Data tag in service class threshold (IN condition) violated monitor element
DATATAGNOTINSC_THRESHOLD_ID	INTEGER	datatagnotinsc_threshold_id - Data tag in service class threshold (NOT IN condition) ID monitor element
DATATAGNOTINSC_THRESHOLD_VALUE	VARCHAR(32)	datatagnotinsc_threshold_value - Data tag in service class threshold (NOT IN condition) value monitor element
DATATAGNOTINSC_THRESHOLD_VIOLATED	SMALLINT	datatagnotinsc_threshold_violated - Data tag in service class threshold (NOT IN condition) violated monitor element
ESTIMATEDSQLCOST_THRESHOLD_ID	INTEGER	estimatedsqlcost_threshold_id - Estimated SQL cost threshold ID monitor element
ESTIMATEDSQLCOST_THRESHOLD_VALUE	BIGINT	estimatedsqlcost_threshold_value - Estimated SQL cost threshold value monitor element
ESTIMATEDSQLCOST_THRESHOLD_VIOLATED	SMALLINT	estimatedsqlcost_threshold_violated - Estimated SQL cost threshold violated monitor element
LAST_REFERENCE_TIME	TIMESTAMP	last_reference_time - Last reference time monitor element
QUERY_DATA_TAG_LIST	VARCHAR(32)	query_data_tag_list - Estimated query data tag list monitor element
QUERY_ACTUAL_DEGREE	INTEGER	query_actual_degree - Actual runtime degree of intrapartition parallelism monitor element
SQLROWSREAD_THRESHOLD_ID	INTEGER	sqlrowsread_threshold_id - SQL rows read threshold ID monitor element
SQLROWSREAD_THRESHOLD_VALUE	BIGINT	sqlrowsread_threshold_value - SQL rows read threshold value monitor element
SQLROWSREAD_THRESHOLD_VIOLATED	SMALLINT	sqlrowsread_threshold_violated - SQL rows read threshold violated monitor element
SQLROWSREADINSC_THRESHOLD_ID	INTEGER	sqlrowsreadinsc_threshold_id - SQL rows read in service class threshold ID monitor element
SQLROWSREADINSC_THRESHOLD_VALUE	BIGINT	sqlrowsreadinsc_threshold_value - SQL rows read in service class threshold value monitor element
SQLROWSREADINSC_THRESHOLD_VIOLATED	SMALLINT	sqlrowsreadinsc_threshold_violated - SQL rows read in service class threshold violated monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
SQLROWSRETURNED_THRESHOLD_ID	INTEGER	sqlrowsreturned_threshold_id - SQL rows read returned threshold ID monitor element
SQLROWSRETURNED_THRESHOLD_VALUE	BIGINT	sqlrowsreturned_threshold_value - SQL rows read returned threshold value monitor element
SQLROWSRETURNED_THRESHOLD_VIOLATED	SMALLINT	sqlrowsreturned_threshold_violated - SQL rows read returned threshold violated monitor element
SQLTEMPSPACE_THRESHOLD_ID	INTEGER	sqltempespace_threshold_id - SQL temporary space threshold ID monitor element
SQLTEMPSPACE_THRESHOLD_VALUE	BIGINT	sqltempespace_threshold_value - SQL temporary space threshold value monitor element
SQLTEMPSPACE_THRESHOLD_VIOLATED	SMALLINT	sqltempespace_threshold_violated - SQL temporary space threshold violated monitor element
WL_WORK_ACTION_SET_ID	INTEGER	wl_work_action_set_id - Workload work action set identifier monitor element
WL_WORK_CLASS_ID	INTEGER	wl_work_class_id - Workload work class identifier monitor element
STMT_TEXT	CLOB(2M)	stmt_text - SQL statement text monitor element
EFF_STMT_TEXT	CLOB(2M)	eff_stmt_text - Effective statement text monitor element
IS_SYSTEM_APPL	SMALLINT	is_system_appl - Is System Application monitor element
NUM_AGENTS	BIGINT	num_agents - Number of Agents Working on a Statement monitor element
AGENTS_TOP	BIGINT	agents_top - Number of Agents Created monitor element
QUERY_CARD_ESTIMATE	BIGINT	query_card_estimate - Query Number of Rows Estimate monitor element
UTILITY_INVOCATION_ID	VARCHAR(32) FOR BIT DATA	utility_invocation_id - Utility invocation ID monitor element
STMT_PKG_CACHE_ID	BIGINT	stmt_pkgcache_id - Statement package cache identifier monitor element
INTRA_PARALLEL_STATE	VARCHAR(3)	intra_parallel_state - Current state of intrapartition parallelism monitor element
NUM_ROUTINES	BIGINT	num_routines - Number of routines monitor element
TOTAL_ACT_TIME	BIGINT	total_act_time - Total activity time monitor element
TOTAL_ACT_WAIT_TIME	BIGINT	total_act_wait_time - Total activity wait time monitor element
POOL_READ_TIME	BIGINT	pool_read_time - Total buffer pool physical read time monitor element
POOL_WRITE_TIME	BIGINT	pool_write_time - Total buffer pool physical write time monitor element
DIRECT_READ_TIME	BIGINT	direct_read_time - Direct read time monitor element
DIRECT_WRITE_TIME	BIGINT	direct_write_time - Direct write time monitor element
LOCK_WAIT_TIME	BIGINT	lock_wait_time - Time waited on locks monitor element
TOTAL_SECTION_SORT_TIME	BIGINT	total_section_sort_time - Total section sort time monitor element
TOTAL_SECTION_SORT_PROC_TIME	BIGINT	total_section_sort_proc_time - Total section sort processing time monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
TOTAL_SECTION_SORTS	BIGINT	total_section_sorts - Total section sorts monitor element
LOCK_ESCALS	BIGINT	lock_escals - Number of lock escalations monitor element
LOCK_WAITS	BIGINT	lock_waits - Lock waits monitor element
ROWS_MODIFIED	BIGINT	rows_modified - Rows modified monitor element
DIRECT_READ_REQS	BIGINT	direct_read_reqs - Direct read requests monitor element
DIRECT_WRITE_REQS	BIGINT	direct_write_reqs - Direct write requests monitor element
POOL_DATA_L_READS	BIGINT	pool_data_l_reads - Buffer pool data logical reads monitor element
POOL_TEMP_DATA_L_READS	BIGINT	pool_temp_data_l_reads - Buffer pool temporary data logical reads monitor element
POOL_XDA_L_READS	BIGINT	pool_xda_l_reads - Buffer pool XDA data logical reads monitor element
POOL_TEMP_XDA_L_READS	BIGINT	pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads monitor element
POOL_INDEX_L_READS	BIGINT	pool_index_l_reads - Buffer pool index logical reads monitor element
POOL_TEMP_INDEX_L_READS	BIGINT	pool_temp_index_l_reads - Buffer pool temporary index logical reads monitor element
POOL_DATA_P_READS	BIGINT	pool_data_p_reads - Buffer pool data physical reads monitor element
POOL_TEMP_DATA_P_READS	BIGINT	pool_temp_data_p_reads - Buffer pool temporary data physical reads monitor element
POOL_XDA_P_READS	BIGINT	pool_xda_p_reads - Buffer pool XDA data physical reads monitor element
POOL_TEMP_XDA_P_READS	BIGINT	pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads monitor element
POOL_INDEX_P_READS	BIGINT	pool_index_p_reads - Buffer pool index physical reads monitor element
POOL_TEMP_INDEX_P_READS	BIGINT	pool_temp_index_p_reads - Buffer pool temporary index physical reads monitor element
POOL_DATA_WRITES	BIGINT	pool_data_writes - Buffer pool data writes monitor element
POOL_XDA_WRITES	BIGINT	pool_xda_writes - Buffer pool XDA data writes monitor element
POOL_INDEX_WRITES	BIGINT	pool_index_writes - Buffer pool index writes monitor element
TOTAL_SORTS	BIGINT	total_sorts - Total sorts monitor element
POST_THRESHOLD_SORTS	BIGINT	post_threshold_sorts - Post threshold sorts monitor element
POST_SHRTHRESHOLD_SORTS	BIGINT	post_shrthreshold_sorts - Post shared threshold sorts monitor element
SORT_OVERFLOWS	BIGINT	sort_overflows - Sort overflows monitor element
WLM_QUEUE_TIME_TOTAL	BIGINT	wlm_queue_time_total - Workload manager total queue time monitor element
WLM_QUEUE_ASSIGNMENTS_TOTAL	BIGINT	wlm_queue_assignments_total - Workload manager total queue assignments monitor element
DEADLOCKS	BIGINT	deadlocks - Deadlocks detected monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
FCM_RECV_VOLUME	BIGINT	fcm_rcv_volume - FCM received volume monitor element
FCM_RECVS_TOTAL	BIGINT	fcm_recvs_total - FCM receives total monitor element
FCM_SEND_VOLUME	BIGINT	fcm_send_volume - FCM send volume monitor element
FCM_SENDS_TOTAL	BIGINT	fcm_sends_total - FCM sends total monitor element
FCM_RECV_WAIT_TIME	BIGINT	fcm_rcv_wait_time - FCM received wait time monitor element
FCM_SEND_WAIT_TIME	BIGINT	fcm_send_wait_time - FCM send wait time monitor element
LOCK_TIMEOUTS	BIGINT	lock_timeouts - Number of lock timeouts monitor element
LOG_BUFFER_WAIT_TIME	BIGINT	log_buffer_wait_time - Log buffer wait time monitor element
NUM_LOG_BUFFER_FULL	BIGINT	num_log_buffer_full - Number of times full log buffer caused agents to wait monitor element
LOG_DISK_WAIT_TIME	BIGINT	log_disk_wait_time - Log disk wait time monitor element
LOG_DISK_WAITS_TOTAL	BIGINT	log_disk_waits_total - Total log disk waits monitor element
TOTAL_ROUTINE_TIME	BIGINT	total_routine_time - Total routine time monitor element
TOTAL_ROUTINE_INVOCATIONS	BIGINT	total_routine_invocations - Total routine invocations monitor elements
COORD_STMT_EXEC_TIME	BIGINT	coord_stmt_exec_time - Execution time for statement by coordinator agent monitor element
STMT_EXEC_TIME	BIGINT	stmt_exec_time - Statement execution time monitor element
TOTAL_SECTION_TIME	BIGINT	total_section_time - Total section time monitor element
TOTAL_SECTION_PROC_TIME	BIGINT	total_section_proc_time - Total section processing time monitor element
TOTAL_ROUTINE_NON_SECT_TIME	BIGINT	total_routine_non_sect_time - Non-section routine execution time monitor elements
TOTAL_ROUTINE_NON_SECT_PROC_TIME	BIGINT	total_routine_non_sect_proc_time - Non-section processing time monitor element
LOCK_WAITS_GLOBAL	BIGINT	lock_waits_global - Lock waits global monitor element
LOCK_WAIT_TIME_GLOBAL	BIGINT	lock_wait_time_global - Lock wait time global monitor element
LOCK_TIMEOUTS_GLOBAL	BIGINT	lock_timeouts_global - Lock timeouts global monitor element
LOCK_ESCALS_MAXLOCKS	BIGINT	lock_escals_maxlocks - Number of maxlocks lock escalations monitor element
LOCK_ESCALS_LOCKLIST	BIGINT	lock_escals_locklist - Number of locklist lock escalations monitor element
LOCK_ESCALS_GLOBAL	BIGINT	lock_escals_global - Number of global lock escalations monitor element
RECLAIM_WAIT_TIME	BIGINT	reclaim_wait_time - Reclaim wait time monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
SPACEMAPPAGE_RECLAIM_WAIT_TIME	BIGINT	spacemappage_reclaim_wait_time - Space map page reclaim wait time monitor element
CF_WAITS	BIGINT	cf_waits - Number of cluster caching facility waits monitor element
CF_WAIT_TIME	BIGINT	cf_wait_time - cluster caching facility wait time monitor element
POOL_DATA_GBP_L_READS	BIGINT	pool_data_gbp_l_reads - Group buffer pool data logical reads monitor element
POOL_DATA_GBP_P_READS	BIGINT	pool_data_gbp_p_reads - Group buffer pool data physical reads monitor element
POOL_DATA_LBP_PAGES_FOUND	BIGINT	pool_data_lbp_pages_found - Local buffer pool found data pages monitor element
POOL_DATA_GBP_INVALID_PAGES	BIGINT	pool_data_gbp_invalid_pages - Group buffer pool invalid data pages monitor element
POOL_INDEX_GBP_L_READS	BIGINT	pool_index_gbp_l_reads - Group buffer pool index logical reads monitor element
POOL_INDEX_GBP_P_READS	BIGINT	pool_index_gbp_p_reads - Group buffer pool index physical reads monitor elements
POOL_INDEX_LBP_PAGES_FOUND	BIGINT	pool_index_lbp_pages_found - Local buffer pool index pages found monitor element
POOL_INDEX_GBP_INVALID_PAGES	BIGINT	pool_index_gbp_invalid_pages - Group buffer pool invalid index pages monitor element
POOL_XDA_GBP_L_READS	BIGINT	pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests monitor element
POOL_XDA_GBP_P_READS	BIGINT	pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests monitor element
POOL_XDA_LBP_PAGES_FOUND	BIGINT	pool_xda_lbp_pages_found - Local buffer pool XDA data pages found monitor element
POOL_XDA_GBP_INVALID_PAGES	BIGINT	pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages monitor element
AUDIT_EVENTS_TOTAL	BIGINT	audit_events_total - Total audit events monitor element
AUDIT_FILE_WRITES_TOTAL	BIGINT	audit_file_writes_total - Total audit files written monitor element
AUDIT_FILE_WRITE_WAIT_TIME	BIGINT	audit_file_write_wait_time - Audit file write wait time monitor element
AUDIT_SUBSYSTEM_WAITS_TOTAL	BIGINT	audit_subsystem_waits_total - Total audit subsystem waits monitor element
AUDIT_SUBSYSTEM_WAIT_TIME	BIGINT	audit_subsystem_wait_time - Audit subsystem wait time monitor element
DIAGLOG_WRITES_TOTAL	BIGINT	diaglog_writes_total - Total diagnostic log file writes monitor element
DIAGLOG_WRITE_WAIT_TIME	BIGINT	diaglog_write_wait_time - Diagnostic log file write wait time monitor element
FCM_MESSAGE_RECVS_TOTAL	BIGINT	fcm_message_recvs_total - Total FCM message receives monitor element
FCM_MESSAGE_RECV_VOLUME	BIGINT	fcm_message_recv_volume - FCM message received volume monitor element
FCM_MESSAGE_RECV_WAIT_TIME	BIGINT	fcm_message_recv_wait_time - FCM message received wait time monitor element
FCM_MESSAGE_SENDS_TOTAL	BIGINT	fcm_message_sends_total - Total FCM message sends monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
FCM_MESSAGE_SEND_VOLUME	BIGINT	fcm_message_send_volume - FCM message send volume monitor element
FCM_MESSAGE_SEND_WAIT_TIME	BIGINT	fcm_message_send_wait_time - FCM message send wait time monitor element
FCM_TQ_RECVS_TOTAL	BIGINT	fcm_tq_recvs_total - FCM table queue receives total monitor element
FCM_TQ_RECV_VOLUME	BIGINT	fcm_tq_recv_volume - FCM table queue received volume monitor element
FCM_TQ_RECV_WAIT_TIME	BIGINT	fcm_tq_recv_wait_time - FCM table queue received wait time monitor element
FCM_TQ_SENDS_TOTAL	BIGINT	fcm_tq_sends_total - FCM table queue send total monitor element
FCM_TQ_SEND_VOLUME	BIGINT	fcm_tq_send_volume - FCM table queue send volume monitor element
FCM_TQ_SEND_WAIT_TIME	BIGINT	fcm_tq_send_wait_time - FCM table queue send wait time monitor element
NUM_LW_THRESH_EXCEEDED	BIGINT	num_lw_thresh_exceeded - Number of lock wait thresholds exceeded monitor element
THRESH_VIOLATIONS	BIGINT	thresh_violations - Number of threshold violations monitor element
TOTAL_APP_SECTION_EXECUTIONS	BIGINT	total_app_section_executions - Total application section executions monitor element
TOTAL_ROUTINE_USER_CODE_PROC_TIME	BIGINT	total_routine_user_code_proc_time - Total routine user code processing time monitor element
TOTAL_ROUTINE_USER_CODE_TIME	BIGINT	total_routine_user_code_time - Total routine user code time monitor element
TQ_TOT_SEND_SPILLS	BIGINT	tq_tot_send_spills - Total number of table queue buffers overflowed monitor element
EVMON_WAIT_TIME	BIGINT	evmon_wait_time - Event monitor wait time monitor element
EVMON_WAITS_TOTAL	BIGINT	evmon_waits_total - Event monitor total waits monitor element
TOTAL_EXTENDED_LATCH_WAIT_TIME	BIGINT	total_extended_latch_wait_time - Total extended latch wait time monitor element
TOTAL_EXTENDED_LATCH_WAITS	BIGINT	total_extended_latch_waits - Total extended latch waits monitor element
TOTAL_DISP_RUN_QUEUE_TIME	BIGINT	total_disp_run_queue_time - Total dispatcher run queue time monitor element
TOTAL_PEDS	BIGINT	total_peds - Total partial early distincts monitor element
DISABLED_PEDS	BIGINT	disabled_peds - Disabled partial early distincts monitor element
POST_THRESHOLD_PEDS	BIGINT	post_threshold_peds - Partial early distincts threshold monitor element
TOTAL_PEAS	BIGINT	total_peas - Total partial early aggregations monitor element
POST_THRESHOLD_PEAS	BIGINT	post_threshold_peas - Partial early aggregation threshold monitor element
TQ_SORT_HEAP_REQUESTS	BIGINT	tq_sort_heap_requests - Table queue sort heap requests monitor element
TQ_SORT_HEAP_REJECTIONS	BIGINT	tq_sort_heap_rejections - Table queue sort heap rejections monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
POOL_QUEUED_ASYNC_DATA_REQS	BIGINT	pool_queued_async_data_reqs - Data prefetch requests monitor element
POOL_QUEUED_ASYNC_INDEX_REQS	BIGINT	pool_queued_async_index_reqs - Index prefetch requests monitor element
POOL_QUEUED_ASYNC_XDA_REQS	BIGINT	pool_queued_async_xda_reqs - XDA prefetch requests monitor element
POOL_QUEUED_ASYNC_TEMP_DATA_REQS	BIGINT	pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element
POOL_QUEUED_ASYNC_TEMP_INDEX_REQS	BIGINT	pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element
POOL_QUEUED_ASYNC_TEMP_XDA_REQS	BIGINT	pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element
POOL_QUEUED_ASYNC_OTHER_REQS	BIGINT	pool_queued_async_other_reqs - Other requests handled by prefetchers monitor element
POOL_QUEUED_ASYNC_DATA_PAGES	BIGINT	pool_queued_async_data_pages - Data pages prefetch requests monitor element
POOL_QUEUED_ASYNC_INDEX_PAGES	BIGINT	pool_queued_async_index_pages - Index pages prefetch requests monitor element
POOL_QUEUED_ASYNC_XDA_PAGES	BIGINT	pool_queued_async_xda_pages - XDA pages prefetch requests monitor element
POOL_QUEUED_ASYNC_TEMP_DATA_PAGES	BIGINT	pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element
POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES	BIGINT	pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element
POOL_QUEUED_ASYNC_TEMP_XDA_PAGES	BIGINT	pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element
POOL_FAILED_ASYNC_DATA_REQS	BIGINT	pool_failed_async_data_reqs - Failed data prefetch requests monitor element
POOL_FAILED_ASYNC_INDEX_REQS	BIGINT	pool_failed_async_index_reqs - Failed index prefetch requests monitor element
POOL_FAILED_ASYNC_XDA_REQS	BIGINT	pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element
POOL_FAILED_ASYNC_TEMP_DATA_REQS	BIGINT	pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element
POOL_FAILED_ASYNC_TEMP_INDEX_REQS	BIGINT	pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element
POOL_FAILED_ASYNC_TEMP_XDA_REQS	BIGINT	pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element
POOL_FAILED_ASYNC_OTHER_REQS	BIGINT	pool_failed_async_other_reqs - Failed non-prefetch requests monitor element
PREFETCH_WAIT_TIME	BIGINT	prefetch_wait_time - Time waited for prefetch monitor element
PREFETCH_WAITS	BIGINT	prefetch_waits - Prefetcher wait count monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP	BIGINT	pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element
POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP	BIGINT	pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element
POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP	BIGINT	pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element
IDA_SEND_WAIT_TIME	BIGINT	ida_send_wait_time - Time spent waiting to send data monitor element
IDA_SENDS_TOTAL	BIGINT	ida_sends_total - Number of times data sent monitor element
IDA_SEND_VOLUME	BIGINT	ida_send_volume - Total data volume sent monitor element
IDA_RECV_WAIT_TIME	BIGINT	ida_recv_wait_time - Time spent waiting to receive data monitor element
IDA_RECVS_TOTAL	BIGINT	ida_recvs_total - Number of times data received monitor element
IDA_RECV_VOLUME	BIGINT	ida_recv_volume - Total data volume received monitor element
ROWS_DELETED	BIGINT	rows_deleted - Rows deleted monitor element
ROWS_INSERTED	BIGINT	rows_inserted - Rows inserted monitor element
ROWS_UPDATED	BIGINT	rows_updated - Rows updated monitor element
TOTAL_HASH_JOINS	BIGINT	total_hash_joins - Total Hash Joins monitor element
TOTAL_HASH_LOOPS	BIGINT	total_hash_loops - Total Hash Loops monitor element
HASH_JOIN_OVERFLOWS	BIGINT	hash_join_overflows - Hash Join Overflows monitor element
HASH_JOIN_SMALL_OVERFLOWS	BIGINT	hash_join_small_overflows - Hash Join Small Overflows monitor element
POST_SHRTHRESHOLD_HASH_JOINS	BIGINT	post_shrthreshold_hash_joins - Post threshold hash joins monitor element
TOTAL_OLAP_FUNCS	BIGINT	total_olap_funcs - Total OLAP Functions monitor element
OLAP_FUNC_OVERFLOWS	BIGINT	olap_func_overflows - OLAP Function Overflows monitor element
INT_ROWS_DELETED	BIGINT	int_rows_deleted - Internal Rows Deleted monitor element
INT_ROWS_INSERTED	BIGINT	int_rows_inserted - Internal Rows Inserted monitor element
INT_ROWS_UPDATED	BIGINT	int_rows_updated - Internal Rows Updated monitor element
POOL_COL_L_READS	BIGINT	pool_col_l_reads - Buffer pool column-organized logical reads monitor element
POOL_TEMP_COL_L_READS	BIGINT	pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads monitor element
POOL_COL_P_READS	BIGINT	pool_col_p_reads - Buffer pool column-organized physical reads monitor element
POOL_TEMP_COL_P_READS	BIGINT	pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
POOL_COL_LBP_PAGES_FOUND	BIGINT	pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found monitor element
POOL_COL_WRITES	BIGINT	pool_col_writes - Buffer pool column-organized writes monitor element
POOL_COL_GBP_L_READS	BIGINT	pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads monitor element
POOL_COL_GBP_P_READS	BIGINT	pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads monitor element
POOL_COL_GBP_INVALID_PAGES	BIGINT	pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages monitor element
POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP	BIGINT	pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool monitor element
POOL_QUEUED_ASYNC_COL_REQS	BIGINT	pool_queued_async_col_reqs - Column-organized prefetch requests monitor element
POOL_QUEUED_ASYNC_TEMP_COL_REQS	BIGINT	pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests monitor element
POOL_QUEUED_ASYNC_COL_PAGES	BIGINT	pool_queued_async_col_pages - Column-organized page prefetch requests monitor element
POOL_QUEUED_ASYNC_TEMP_COL_PAGES	BIGINT	pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests monitor element
POOL_FAILED_ASYNC_COL_REQS	BIGINT	pool_failed_async_col_reqs - Failed column-organized prefetch requests monitor element
POOL_FAILED_ASYNC_TEMP_COL_REQS	BIGINT	pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests monitor element
TOTAL_COL_TIME	BIGINT	total_col_time - Total column-organized time monitor element
TOTAL_COL_PROC_TIME	BIGINT	total_col_proc_time - Total column-organized processing time monitor element
TOTAL_COL_EXECUTIONS	BIGINT	total_col_executions - Total column-organized executions monitor element
COMM_EXIT_WAIT_TIME	BIGINT	comm_exit_wait_time - Communication buffer exit wait time
COMM_EXIT_WAITS	BIGINT	comm_exit_waits - Communication buffer exit number of waits
POST_THRESHOLD_HASH_JOINS	BIGINT	post_threshold_hash_joins - Hash Join Threshold monitor element
POOL_DATA_CACHING_TIER_L_READS	BIGINT	Reserved for future use.
POOL_INDEX_CACHING_TIER_L_READS	BIGINT	Reserved for future use.
POOL_XDA_CACHING_TIER_L_READS	BIGINT	Reserved for future use.
POOL_COL_CACHING_TIER_L_READS	BIGINT	Reserved for future use.

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
POOL_DATA_CACHING_TIER_PAGE_WRITES	BIGINT	Reserved for future use.
POOL_INDEX_CACHING_TIER_PAGE_WRITES	BIGINT	Reserved for future use.
POOL_XDA_CACHING_TIER_PAGE_WRITES	BIGINT	Reserved for future use.
POOL_COL_CACHING_TIER_PAGE_WRITES	BIGINT	Reserved for future use.
POOL_DATA_CACHING_TIER_PAGE_UPDATES	BIGINT	Reserved for future use.
POOL_INDEX_CACHING_TIER_PAGE_UPDATES	BIGINT	Reserved for future use.
POOL_XDA_CACHING_TIER_PAGE_UPDATES	BIGINT	Reserved for future use.
POOL_COL_CACHING_TIER_PAGE_UPDATES	BIGINT	Reserved for future use.
POOL_CACHING_TIER_PAGE_READ_TIME	BIGINT	Reserved for future use.
POOL_CACHING_TIER_PAGE_WRITE_TIME	BIGINT	Reserved for future use.
POOL_DATA_CACHING_TIER_PAGES_FOUND	BIGINT	Reserved for future use.
POOL_INDEX_CACHING_TIER_PAGES_FOUND	BIGINT	Reserved for future use.
POOL_XDA_CACHING_TIER_PAGES_FOUND	BIGINT	Reserved for future use.
POOL_COL_CACHING_TIER_PAGES_FOUND	BIGINT	Reserved for future use.
POOL_DATA_CACHING_TIER_GBP_INVALID_PAGES	BIGINT	Reserved for future use.
POOL_INDEX_CACHING_TIER_GBP_INVALID_PAGES	BIGINT	Reserved for future use.
POOL_XDA_CACHING_TIER_GBP_INVALID_PAGES	BIGINT	Reserved for future use.
POOL_COL_CACHING_TIER_GBP_INVALID_PAGES	BIGINT	Reserved for future use.
POOL_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND	BIGINT	Reserved for future use.
POOL_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND	BIGINT	Reserved for future use.
POOL_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND	BIGINT	Reserved for future use.
POOL_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND	BIGINT	Reserved for future use.
TOTAL_HASH_GRPBYS	BIGINT	total_hash_grpbys - Total hash group by operations
HASH_GRPBY_OVERFLOWS	BIGINT	hash_grpby_overflows - Hash group by overflows
POST_THRESHOLD_HASH_GRPBYS	BIGINT	post_threshold_hash_grpbys - Hash group by threshold
POST_THRESHOLD_OLAP_FUNCS	BIGINT	post_threshold_olap_funcs - OLAP function threshold monitor element
EXT_TABLE_RECV_WAIT_TIME	BIGINT	ext_table_recv_wait_time - Total agent wait time for external table readers monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
EXT_TABLE_RECVS_TOTAL	BIGINT	ext_table_recvs_total - Total row batches received from external table readers monitor element
EXT_TABLE_RECV_VOLUME	BIGINT	ext_table_recv_volume - Total data received from external table readers monitor element
EXT_TABLE_READ_VOLUME	BIGINT	ext_table_read_volume - Total data read by external table readers monitor element
EXT_TABLE_SEND_WAIT_TIME	BIGINT	ext_table_send_wait_time - Total agent wait time for external table writers monitor element
EXT_TABLE_SENDS_TOTAL	BIGINT	ext_table_sends_total - Total row batches sent to external table writers monitor element
EXT_TABLE_SEND_VOLUME	BIGINT	ext_table_send_volume - Total data sent to external table writers monitor element
EXT_TABLE_WRITE_VOLUME	BIGINT	ext_table_write_volume - Total data written by external table writers monitor element
SEMANTIC_ENV_ID	BIGINT	semantic_env_id - Query semantic compilation environment ID monitor element
STMTID	BIGINT	stmtid - Query statement ID monitor element
PLANID	BIGINT	planid - Query plan ID monitor element
POST_THRESHOLD_COL_VECTOR_CONSUMERS	BIGINT	post_threshold_col_vector_consumers - Post threshold columnar_vector consumers
TOTAL_COL_VECTOR_CONSUMERS	BIGINT	total_col_vector_consumers - Total columnar_vector consumers
ACTIVE_HASH_GRPBY	BIGINT	active_hash_grpbys - Active hash GROUP BY operations monitor element
ACTIVE_HASH_GRPBY_TOP	BIGINT	active_hash_grpbys_top - Active hash Group By operations high watermark
ACTIVE_HASH_JOINS	BIGINT	active_hash_joins - Active hash joins monitor element
ACTIVE_HASH_JOINS_TOP	BIGINT	active_hash_joins_top - Active hash joins operations high watermark
ACTIVE_OLAP_FUNCS	BIGINT	active_olap_funcs - Active OLAP Functions monitor element
ACTIVE_OLAP_FUNCS_TOP	BIGINT	active_olap_funcs_top - Active OLAP functions operations high watermark
ACTIVE_PEAS	BIGINT	active_peas - Active partial early aggregations operations
ACTIVE_PEAS_TOP	BIGINT	active_peas_top - Active partial early aggregations operations high watermark
ACTIVE_PEDS	BIGINT	active_peds - Active partial early distinct operations
ACTIVE_PEDS_TOP	BIGINT	active_peds_top - Active partial early distinct operations high watermark
ACTIVE_SORT_CONSUMERS	BIGINT	active_sort_consumers - Active sort consumers
ACTIVE_SORT_CONSUMERS_TOP	BIGINT	active_sort_consumers_top - Active sort consumers high watermark
ACTIVE_SORTS	BIGINT	active_sorts - Active Sorts monitor element
ACTIVE_SORTS_TOP	BIGINT	active_sorts_top - Active Sorts high watermark
ACTIVE_COL_VECTOR_CONSUMERS	BIGINT	active_col_vector_consumers - Active columnar_vector consumers
ACTIVE_COL_VECTOR_CONSUMERS_TOP	BIGINT	active_col_vector_consumers_top - Active columnar_vector consumers high watermark

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
SORT_CONSUMER_HEAP_TOP	BIGINT	sort_consumer_heap_top - Individual private sort heap consumer high watermark
SORT_CONSUMER_SHRHEAP_TOP	BIGINT	sort_consumer_shrheap_top - Individual shared sort heap consumer high watermark
SORT_HEAP_ALLOCATED	BIGINT	sort_heap_allocated - Total Sort Heap Allocated monitor element
SORT_HEAP_TOP	BIGINT	sort_heap_top - Sort private heap high watermark monitor element
SORT_SHRHEAP_ALLOCATED	BIGINT	sort_shrheap_allocated - Sort Share Heap Currently Allocated monitor element
SORT_SHRHEAP_TOP	BIGINT	sort_shrheap_top - Sort share heap high watermark monitor element
TOTAL_INDEX_BUILD_TIME	BIGINT	total_index_build_time - Total time spent building indexes due to index creation
TOTAL_INDEX_BUILD_PROC_TIME	BIGINT	total_index_build_proc_time - Total non-wait time spent building indexes due to index creation
TOTAL_INDEXES_BUILT	BIGINT	total_indexes_built - Total number of indexes built
FCM_TQ_RECV_WAITS_TOTAL	BIGINT	fcm_tq_rcv_waits_total - Number of times spent waiting to receive the next buffer monitor element
FCM_MESSAGE_RECV_WAITS_TOTAL	BIGINT	fcm_message_rcv_waits_total - Number of times spent waiting for FCM reply message monitor element
FCM_TQ_SEND_WAITS_TOTAL	BIGINT	fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element
FCM_MESSAGE_SEND_WAITS_TOTAL	BIGINT	fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element
FCM_SEND_WAITS_TOTAL	BIGINT	fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element
FCM_RECV_WAITS_TOTAL	BIGINT	fcm_rcv_waits_total - Number of times spent waiting to receive data through FCM monitor element
ACTIVITYTOTALRUNTIME_THRESHOLD_ID	INTEGER	activitytotalruntime_threshold_id - Activity total runtime threshold ID monitor element
ACTIVITYTOTALRUNTIME_THRESHOLD_VALUE	TIMESTAMP	activitytotalruntime_threshold_value - Activity total runtime threshold value monitor element
ACTIVITYTOTALRUNTIME_THRESHOLD_VIOLATED	SMALLINT	activitytotalruntime_threshold_violated - Activity total runtime threshold violated monitor element
ACTIVITYTOTALRUNTIMEINALLSC_THRESHOLD_ID	INTEGER	activitytotalruntimeinallsc_threshold_id - Activity total runtime in all service classes threshold ID monitor element
ACTIVITYTOTALRUNTIMEINALLSC_THRESHOLD_VALUE	TIMESTAMP	activitytotalruntimeinallsc_threshold_value - Activity total runtime in all service classes threshold value monitor element
ACTIVITYTOTALRUNTIMEINALLSC_THRESHOLD_VIOLATED	SMALLINT	activitytotalruntimeinallsc_threshold_violated - Activity total runtime in all service classes threshold violated monitor element
ADM_BYPASSED	SMALLINT	Reserved for future use.adm_bypassed - Admission control bypassed monitor element
COL_VECTOR_CONSUMER_OVERFLOW	BIGINT	col_vector_consumer_overflows - Columnar vector consumer overflows monitor element.
TOTAL_COL_SYNOPSIS_TIME	BIGINT	total_col_synopsis_time - Total column-organized synopsis table time monitor element

Table 107. Information returned by the MON\_GET\_ACTIVITY routine (continued)

Column name	Data type	Description
TOTAL_COL_SYNOPSIS_PROC_TIME	BIGINT	total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element
TOTAL_COL_SYNOPSIS_EXECUTIONS	BIGINT	total_col_synopsis_executions - Total column-organized synopsis table executions monitor element
COL_SYNOPSIS_ROWS_INSERTED	BIGINT	col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element
ADM_OVERFLOWS	BIGINT	Reserved for future use.
ADM_BYPASS_ACT_TOTAL	BIGINT	Reserved for future use.
LOB_PREFETCH_WAIT_TIME	BIGINT	lob_prefetch_wait_time - LOB prefetch wait time monitor element
LOB_PREFETCH_REQS	BIGINT	lob_prefetch_reqs - LOB prefetcher request count monitor element
FED_ROWS_DELETED	BIGINT	fed_rows_deleted - Rows deleted by a federation system monitor element
FED_ROWS_INSERTED	BIGINT	fed_rows_inserted - Rows inserted by a federation system monitor element
FED_ROWS_UPDATED	BIGINT	fed_rows_updated - Rows updated by a federation system monitor element
FED_ROWS_READ	BIGINT	fed_rows_read - Rows read by a federation system monitor element
FED_WAIT_TIME	BIGINT	fed_wait_time - Time spent by a federation server monitor element
FED_WAITS_TOTAL	BIGINT	fed_waits_total - Total number of execution times for a federation server monitor element
ESTIMATED_SORT_SHRHEAP_TOP	BIGINT	Reserved for future use.
ESTIMATED_SORT_CONSUMERS_TOP	BIGINT	Reserved for future use.
ESTIMATED_RUNTIME	BIGINT	Reserved for future use.

## Examples

### Activities currently running with a known application handle

After you identify the application handle, you can look up all the activities currently running in this application. For example, suppose that an administrator wants to list the activities of an application whose application handle, determined by using the **LIST APPLICATIONS** command, is 1. The administrator runs the following query:

```
SELECT SUBSTR(CHAR(COORD_MEMBER),1,5) AS COORD,
 SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
 SUBSTR(CHAR(UOW_ID),1,5) AS UOWID,
 SUBSTR(CHAR(ACTIVITY_ID),1,5) AS ACTID,
 SUBSTR(CHAR(PARENT_UOW_ID),1,8) AS PARUOWID,
 SUBSTR(CHAR(PARENT_ACTIVITY_ID),1,8) AS PRACTID,
 ACTIVITY_TYPE AS ACTTYPE,
 SUBSTR(CHAR(NESTING_LEVEL),1,7) AS NESTING
FROM TABLE(MON_GET_ACTIVITY(1, -2)) AS WLOACTS
ORDER BY MEMB, UOWID, ACTID
```

Sample output from the query is as follows:

```
COORD MEMB UOWID ACTID PARUOWID PRACTID ACTTYPE NESTING

0 0 2 3 - - CALL 0
0 0 2 5 2 3 READ_DML 1
```

## Activities currently running on the system

The following query uses MON\_GET\_ACTIVITY to provide statement text for all SQL activities currently running on the system:

```
SELECT t.application_handle, t.uow_id, t.activity_id,
 VARCHAR(t.stmt_text, 256) as stmt_text
FROM table(MON_GET_ACTIVITY(NULL, -1)) as t
```

Sample output is as follows:

APPLICATION_HANDLE	UOW_ID	ACTIVITY_ID	STMT_TEXT
1	1	1	SELECT * FROM SYSCAT.TABLES
47	1	36	INSERT INTO T1 VALUES(123)

## MON\_GET\_ACTIVITY\_DETAILS table function - Return information about an activity as an XML document

The MON\_GET\_ACTIVITY\_DETAILS table function returns information about an activity, including general activity information (like statement text) and a set of metrics for the activity.

The metrics are returned in an XML document.

## Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

## Default PUBLIC privilege

None

## Syntax

```
►► MON_GET_ACTIVITY_DETAILS ((—application_handle—, —uow_id—, —
► activity_id—, —member—)) ►►
```

The schema is SYSPROC.

## Routine parameters

### *application\_handle*

An input argument of type BIGINT that specifies a valid application handle. If the argument is null, no rows are returned from this function, and an SQL0171N error is returned.

### *uow\_id*

An input argument of type INTEGER that specifies a valid unit of work identifier unique within the application. If the argument is null, no rows are returned from this function, and an SQL0171N error is returned.

### *activity\_id*

An input argument of type INTEGER that specifies a valid activity ID unique

within the unit of work. If the argument is null, no rows are returned from this function, and an SQL0171N error is returned.

**member**

An input argument of type INTEGER that specifies a valid member number in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

**Information returned**

*Table 108. Information returned for MON\_GET\_ACTIVITY\_DETAILS*

Column name	Data type	Description
APPLICATION_HANDLE	BIGINT	application_handle - Application handle
UOW_ID	INTEGER	uow_id - Unit of work ID
ACTIVITY_ID	INTEGER	activity_id - Activity ID
MEMBER	SMALLINT	member - Database member
DETAILS	BLOB(8M)	XML document that contains activity details. See Table 109 for a description of the elements in this document.

The following example shows the structure of the XML document that is returned in the DETAILS column.

```
<db2_activity_details xmlns="http://www.ibm.com/xmlns/prod/db2/mon" release="90700000">
 <member>0</member>
 <application_handle>70</application_handle>
 <activity_id>1</activity_id>
 <activity_state>IDLE</activity_state>
 <activity_type>READ_DML</activity_type>
 <uow_id>1</uow_id>
 ...
 <activity_metrics release="90700000">
 <lock_wait_time>2000</lock_wait_time>
 ...
 </activity_metrics>
</db2_activity_details>
```

For the full schema, see `sqllib/misc/DB2MonRoutines.xsd`. This document uses the following XML non-primitive type definitions:

```
<xs:simpleType name = "executable_id_type" >
 <xs:annotation>
 <xs:documentation>
 The binary Executable ID
 </xs:documentation>
 </xs:annotation>
 <xs:restriction base = "xs:hexBinary" >
 <xs:maxLength value = "32" />
 </xs:restriction>
</xs:simpleType>
```

**Detailed metrics returned**

*Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS*

Element name	Data type	Description or corresponding monitor element
active_col_vector_consumers	xs:long	active_col_vector_consumers - Active columnar_vector consumers

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
active_col_vector_consumers_top	xs:long	active_col_vector_consumers_top - Active columnar_vector consumers high watermark
active_hash_grpbys	xs:long	active_hash_grpbys - Active hash GROUP BY operations monitor element
active_hash_grpbys_top	xs:long	active_hash_grpbys_top - Active hash Group By operations high watermark
active_hash_joins	xs:long	active_hash_joins - Active hash joins monitor element
active_hash_joins_top	xs:long	active_hash_joins_top - Active hash joins operations high watermark
active_olap_funcs	xs:long	active_olap_funcs - Active OLAP Functions monitor element
active_olap_funcs_top	xs:long	active_olap_funcs_top - Active OLAP functions operations high watermark
active_peas	xs:long	active_peas - Active partial early aggregations operations
active_peas_top	xs:long	active_peas_top - Active partial early aggregations operations high watermark
active_peds	xs:long	active_peds - Active partial early distinct operations
active_peds_top	xs:long	active_peds_top - Active partial early distinct operations high watermark
active_sort_consumers	xs:long	active_sort_consumers - Active sort consumers
active_sort_consumers_top	xs:long	active_sort_consumers_top - Active sort consumers high watermark
active_sorts	xs:long	active_sorts - Active Sorts monitor element
active_sorts_top	xs:long	active_sorts_top - Active Sorts high watermark
activity_id	xs:nonNegativeInteger	activity_id - Activity ID
activity_state	xs:string	activity_state - Activity state
activity_type	xs:string	activity_type - Activity type
activitytotalruntime_threshold_id	xs:int	activitytotalruntime_threshold_id - Activity total runtime threshold ID monitor element
activitytotalruntime_threshold_value	xs:dateTime	activitytotalruntime_threshold_value - Activity total runtime threshold value monitor element
activitytotalruntime_threshold_violated	xs:boolean	activitytotalruntime_threshold_violated - Activity total runtime threshold violated monitor element
activitytotalruntimeinallsc_threshold_id	xs:int	activitytotalruntimeinallsc_threshold_id - Activity total runtime in all service classes threshold ID monitor element
activitytotalruntimeinallsc_threshold_value	xs:dateTime	activitytotalruntimeinallsc_threshold_value - Activity total runtime in all service classes threshold value monitor element
activitytotalruntimeinallsc_threshold_violated	xs:boolean	activitytotalruntimeinallsc_threshold_violated - Activity total runtime in all service classes threshold violated monitor element
activitytotaltime_threshold_id	xs:int	activitytotaltime_threshold_id - Activity total time threshold ID
activitytotaltime_threshold_value	xs:dateTime	activitytotaltime_threshold_value - Activity total time threshold value

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
activitytotaltime_threshold_violated	xs:short (1 = yes, 0 = no)	activitytotaltime_threshold_violated - Activity total time threshold violated
adm_bypassed	xs:short	Reserved for future use.adm_bypassed - Admission control bypassed monitor element
adm_bypass_act_total	xs:long	Reserved for future use.
adm_overflows	xs:long	Reserved for future use.
agents_top	xs:long	agents_top - Number of Agents Created monitor element
aggsqltempespace_threshold_id	xs:int	aggsqltempespace_threshold_id - AggSQL temporary space threshold ID.
aggsqltempespace_threshold_value	xs:long	aggsqltempespace_threshold_value - AggSQL temporary space threshold value
aggsqltempespace_threshold_violated	xs:short (1 = yes, 0 = no)	aggsqltempespace_threshold_violated - AggSQL temporary space threshold violated
application_handle	xs:nonNegativeInteger	application_handle - Application handle
audit_events_total	xs:nonNegativeInteger	audit_events_total - Total audit events
audit_file_write_wait_time	xs:nonNegativeInteger	audit_file_write_wait_time - Audit file write wait time
audit_file_writes_total	xs:nonNegativeInteger	audit_file_writes_total - Total Audit files written
audit_subsystem_wait_time	xs:nonNegativeInteger	audit_subsystem_wait_time - Audit subsystem wait time
audit_subsystem_waits_total	xs:nonNegativeInteger	audit_subsystem_waits_total - Total audit subsystem waits
cf_wait_time	xs:long	cf_wait_time - cluster caching facility wait time monitor element
cf_waits	xs:long	cf_waits - Number of cluster caching facility waits monitor element
client_acctng	xs:string(255)	client_acctng - Client accounting string monitor element
client_applname	xs:string(255)	client_applname - Client application name monitor element
client_userid	xs:string(255)	client_userid - Client user ID monitor element
client_wrkstnname	xs:string(255)	client_wrkstnname - Client workstation name monitor element
col_synopsis_rows_inserted	xs:nonNegativeInteger	col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element
col_vector_consumer_overflows	xs:long	col_vector_consumer_overflows - Columnar vector consumer overflows monitor element.
comm_exit_wait_time	xs:long	comm_exit_wait_time - Communication exit wait time monitor element
comm_exit_waits	xs:long	comm_exit_waits - Communication exit number of waits monitor element
concurrentdbcoordactivities_db_threshold_id	xs:int	concurrentdbcoordactivities_db_threshold_id - Concurrent database coordinator activities threshold ID
concurrentdbcoordactivities_db_threshold_queued	xs:short (1 = yes, 0 = no)	concurrentdbcoordactivities_db_threshold_queued - Concurrent database coordinator activities
concurrentdbcoordactivities_db_threshold_value	xs:long	concurrentdbcoordactivities_db_threshold_value - Concurrent database coordinator activities

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
concurrentdbcoordactivities_db_threshold_violated	xs:short (1 = yes, 0 = no)	concurrentdbcoordactivities_db_threshold_violated - Concurrent database coordinator activities threshold violated
concurrentdbcoordactivities_subclass_threshold_id	xs:int	concurrentdbcoordactivities_subclass_threshold_id - Concurrent database coordinator activities subclass threshold ID
concurrentdbcoordactivities_subclass_threshold_queued	xs:short (1 = yes, 0 = no)	concurrentdbcoordactivities_subclass_threshold_queued - Concurrent database coordinator activities subclass threshold queued
concurrentdbcoordactivities_subclass_threshold_value	xs:long	concurrentdbcoordactivities_subclass_threshold_value - Concurrent database coordinator activities subclass threshold value
concurrentdbcoordactivities_subclass_threshold_violated	xs:short (1 = yes, 0 = no)	concurrentdbcoordactivities_subclass_threshold_violated - Concurrent database coordinator activities subclass threshold violated
concurrentdbcoordactivities_superclass_threshold_id	xs:int	concurrentdbcoordactivities_superclass_threshold_id - Concurrent database coordinator activities superclass
concurrentdbcoordactivities_superclass_threshold_queued	xs:short (1 = yes, 0 = no)	concurrentdbcoordactivities_superclass_threshold_queued - Concurrent database coordinator activities superclass threshold queued
concurrentdbcoordactivities_superclass_threshold_value	xs:long	concurrentdbcoordactivities_superclass_threshold_value - Concurrent database coordinator activities superclass threshold value
concurrentdbcoordactivities_superclass_threshold_violated	xs:short (1 = yes, 0 = no)	concurrentdbcoordactivities_superclass_threshold_violated - Concurrent database coordinator activities superclass threshold violated
concurrentdbcoordactivities_wl_was_threshold_id	xs:int	concurrentdbcoordactivities_wl_was_threshold_id - Concurrent database coordinator activities workload work action set threshold ID
concurrentdbcoordactivities_wl_was_threshold_queued	xs:short (1 = yes, 0 = no)	concurrentdbcoordactivities_wl_was_threshold_queued - Concurrent database coordinator activities workload work action set threshold queued
concurrentdbcoordactivities_wl_was_threshold_value	xs:long	concurrentdbcoordactivities_wl_was_threshold_value - Concurrent database coordinator activities workload work action set threshold value
concurrentdbcoordactivities_wl_was_threshold_violated	xs:short (1 = yes, 0 = no)	concurrentdbcoordactivities_wl_was_threshold_violated - Concurrent database coordinator activities workload work action set threshold violated
concurrentdbcoordactivities_work_action_set_threshold_id	xs:int	concurrentdbcoordactivities_work_action_set_threshold_id - Concurrent database coordinator activities work action set threshold ID
concurrentdbcoordactivities_work_action_set_threshold_queued	xs:short (1 = yes, 0 = no)	concurrentdbcoordactivities_work_action_set_threshold_queued - Concurrent database coordinator activities work action set threshold queued
concurrentdbcoordactivities_work_action_set_threshold_value	xs:long	concurrentdbcoordactivities_work_action_set_threshold_value - Concurrent database coordinator activities work action set threshold value

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
concurrentdbcoordactivities_work_action_set_threshold_violated	xs:short (1 = yes, 0 = no)	concurrentdbcoordactivities_work_action_set_threshold_violated - Concurrent database coordinator activities work action set threshold violated
concurrentworkloadactivities_threshold_id	xs:int	concurrentworkloadactivities_threshold_id - Concurrent workload activities threshold ID monitor element
concurrentworkloadactivities_threshold_value	xs:long	concurrentworkloadactivities_threshold_value - Concurrent workload activities threshold value monitor element
concurrentworkloadactivities_threshold_violated	xs:short	concurrentworkloadactivities_threshold_violated - Concurrent workload activities threshold violated monitor element
coord_member	xs:nonNegativeInteger	coord_member - Coordinator member
coord_stmt_exec_time	xs:nonNegativeInteger	coord_stmt_exec_time - Execution time for statement by coordinator agent
cputime_threshold_id	xs:int	cputime_threshold_id - CPU time threshold ID
cputime_threshold_value	xs:long	cputime_threshold_value - CPU time threshold value
cputime_threshold_violated	xs:short (1 = yes, 0 = no)	cputime_threshold_violated - CPU time threshold violated
cputimeinsc_threshold_id	xs:int	cputimeinsc_threshold_id - CPU time in service threshold ID
cputimeinsc_threshold_value	xs:long	cputimeinsc_threshold_value - CPU time in service threshold value
cputimeinsc_threshold_violated	xs:short (1 = yes, 0 = no)	cputimeinsc_threshold_violated - CPU time in service threshold violated
database_work_action_set_id	xs:nonNegativeInteger	db_work_action_set_id - Database work action set ID
database_work_class_id	xs:nonNegativeInteger	db_work_class_id - Database work class ID
datataginsc_threshold_id	xs:int	datataginsc_threshold_id - Datataginsc in threshold identifier
datataginsc_threshold_value	xs:string(32)	datataginsc_threshold_value - Datataginsc in threshold value
datataginsc_threshold_violated	xs:short (1 = yes, 0 = no)	datataginsc_threshold_violated - Datataginsc in threshold violated
datatagnotinsc_threshold_id	xs:int	datatagnotinsc_threshold_id - Datatagnotinsc not in threshold identifier
datatagnotinsc_threshold_value	xs:string(32)	datatagnotinsc_threshold_value - Datatagnotinsc not in threshold value
datatagnotinsc_threshold_violated	xs:short (1 = yes, 0 = no)	datatagnotinsc_threshold_violated - Datatagnotinsc not in threshold violated
deadlocks	xs:nonNegativeInteger	deadlocks - Deadlocks detected
diaglog_write_wait_time	xs:nonNegativeInteger	diaglog_write_wait_time - Diag log write time
diaglog_writes_total	xs:nonNegativeInteger	diaglog_writes_total - Diag log total writes
direct_read_reqs	xs:nonNegativeInteger	direct_read_reqs - Direct read requests
direct_read_time	xs:nonNegativeInteger	direct_read_time - Direct read time
direct_reads	xs:nonNegativeInteger	direct_reads - Direct reads from database
direct_write_reqs	xs:nonNegativeInteger	direct_write_reqs - Direct write requests

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
direct_write_time	xs:nonNegativeInteger	direct_write_time - Direct write time
direct_writes	xs:nonNegativeInteger	direct_writes - Direct writes to database
disabled_peds	xs:long	disabled_peds - Disabled partial early distincts monitor element
eff_stmt_text	xs:string	eff_stmt_text - Effective statement text. The first 1024 characters of the concentrated statement text following any literal replacement done by the statement concentrator. Only present if the statement concentrator is enabled and this statement was altered by the statement concentrator.
effective_isolation	xs:string	effective_isolation - Effective isolation
effective_lock_timeout	xs:integer	effective_lock_timeout - Effective lock time-out
effective_query_degree	xs:integer	effective_query_degree - Effective query degree
entry_time	xs:dateTime	entry_time - Entry timeThe time that this activity arrived into the system.
estimated_runtime	xs:long	Reserved for future use.
estimated_sort_consumers_top	xs:long	Reserved for future use.
estimated_sort_shrheap_top	xs:long	Reserved for future use.
estimatedsqlcost_threshold_id	xs:int	estimatedsqlcost_threshold_id - Estimated SQL cost threshold ID
estimatedsqlcost_threshold_value	xs:long	estimatedsqlcost_threshold_value - Estimated SQL cost threshold value
estimatedsqlcost_threshold_violated	xs:short (1 = yes, 0 = no)	estimatedsqlcost_threshold_violated - Estimated SQL cost threshold violated
evmon_wait_time	xs:nonNegativeInteger	evmon_wait_time - Event monitor wait time
evmon_waits_total	xs:nonNegativeInteger	evmon_waits_total - Event monitor total waits
executable_id	executable_id_type	executable_id - Executable ID
ext_table_read_volume	xs:nonNegativeInteger	ext_table_read_volume - Total data read by external table readers monitor element
ext_table_rcv_volume	xs:nonNegativeInteger	ext_table_rcv_volume - Total data received from external table readers monitor element
ext_table_rcv_wait_time	xs:nonNegativeInteger	ext_table_rcv_wait_time - Total agent wait time for external table readers monitor element
ext_table_recvs_total	xs:nonNegativeInteger	ext_table_recvs_total - Total row batches received from external table readers monitor element
ext_table_send_volume	xs:nonNegativeInteger	ext_table_send_volume - Total data sent to external table writers monitor element
ext_table_send_wait_time	xs:nonNegativeInteger	ext_table_send_wait_time - Total agent wait time for external table writers monitor element
ext_table_sends_total	xs:nonNegativeInteger	ext_table_sends_total - Total row batches sent to external table writers monitor element
ext_table_write_volume	xs:nonNegativeInteger	ext_table_write_volume - Total data written by external table writers monitor element
fcm_message_rcv_volume	xs:nonNegativeInteger	fcm_message_rcv_volume - FCM message rcv volume
fcm_message_rcv_wait_time	xs:nonNegativeInteger	fcm_message_rcv_wait_time - FCM message rcv wait time

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
fcm_message_rcv_waits_total	xs:nonNegativeInteger	fcm_message_rcv_waits_total - Number of times spent waiting for FCM reply message monitor element
fcm_message_rcvs_total	xs:nonNegativeInteger	fcm_message_rcvs_total - FCM message rcvs total
fcm_message_send_volume	xs:nonNegativeInteger	fcm_message_send_volume - FCM message send volume
fcm_message_send_wait_time	xs:nonNegativeInteger	fcm_message_send_wait_time - FCM message send wait time
fcm_message_send_waits_total	xs:nonNegativeInteger	fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element
fcm_message_sends_total	xs:nonNegativeInteger	fcm_message_sends_total - FCM message sends total
fcm_rcv_volume	xs:nonNegativeInteger	fcm_rcv_volume - FCM rcv volume
fcm_rcv_wait_time	xs:nonNegativeInteger	fcm_rcv_wait_time - FCM rcv wait time
fcm_rcv_waits_total	xs:nonNegativeInteger	fcm_rcv_waits_total - Number of times spent waiting to receive data through FCM monitor element
fcm_rcvs_total	xs:nonNegativeInteger	fcm_rcvs_total - FCM rcvs total
fcm_send_volume	xs:nonNegativeInteger	fcm_send_volume - FCM send volume
fcm_send_wait_time	xs:nonNegativeInteger	fcm_send_wait_time - FCM send wait time
fcm_send_waits_total	xs:nonNegativeInteger	fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element
fcm_sends_total	xs:nonNegativeInteger	fcm_sends_total - FCM sends total
fcm_tq_rcv_volume	xs:nonNegativeInteger	fcm_tq_rcv_volume - FCM tablequeue rcv volume
fcm_tq_rcv_wait_time	xs:nonNegativeInteger	fcm_tq_rcv_wait_time - FCM tablequeue rcv wait time
fcm_tq_rcv_waits_total	xs:nonNegativeInteger	fcm_tq_rcv_waits_total - Number of times spent waiting to receive the next buffer monitor element
fcm_tq_rcvs_total	xs:nonNegativeInteger	fcm_tq_rcvs_total - FCM tablequeue rcvs total
fcm_tq_send_volume	xs:nonNegativeInteger	fcm_tq_send_volume - FCM tablequeue send volume
fcm_tq_send_wait_time	xs:nonNegativeInteger	fcm_tq_send_wait_time - FCM tablequeue send wait time
fcm_tq_send_waits_total	xs:nonNegativeInteger	fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element
fcm_tq_sends_total	xs:nonNegativeInteger	fcm_tq_sends_total - FCM tablequeue send total
fed_rows_deleted	xs:nonNegativeInteger	fed_rows_deleted - Rows deleted by a federation system monitor element
fed_rows_inserted	xs:nonNegativeInteger	fed_rows_inserted - Rows inserted by a federation system monitor element
fed_rows_read	xs:nonNegativeInteger	fed_rows_read - Rows read by a federation system monitor element
fed_rows_updated	xs:nonNegativeInteger	fed_rows_updated - Rows updated by a federation system monitor element
fed_wait_time	xs:nonNegativeInteger	fed_wait_time - Time spent by a federation server monitor element
fed_waits_total	xs:nonNegativeInteger	fed_waits_total - Total number of execution times for a federation server monitor element
hash_grpby_overflows	xs:long	hash_grpby_overflows - Hash group by overflows
hash_join_overflows	xs:long	hash_join_overflows - Hash Join Overflows monitor element

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
hash_join_small_overflows	xs:long	hash_join_small_overflows - Hash Join Small Overflows monitor element
ida_rcv_volume	xs:nonNegativeInteger	ida_rcv_volume - Total data volume received monitor element  The total volume of data the database server received from an in-database analytics process. The value is reported in bytes.
ida_rcv_wait_time	xs:nonNegativeInteger	ida_rcv_wait_time - Time spent waiting to receive data monitor element  The total amount of time spent waiting to receive data from an in-database analytics process.
ida_rcvs_total	xs:nonNegativeInteger	ida_rcvs_total - Number of times data received monitor element  The total number of times data was received from an in-database analytics process.
ida_send_volume	xs:nonNegativeInteger	ida_send_volume - Total data volume sent monitor element  The total volume of data sent from the database server to an in-database analytics process. The value is reported in bytes.
ida_send_wait_time	xs:nonNegativeInteger	ida_send_wait_time - Time spent waiting to send data monitor element  The total amount of time spent waiting to send data to an in-database analytics process.
ida_sends_total	xs:nonNegativeInteger	ida_sends_total - Number of times data sent monitor element  The total number of times data was sent to an in-database analytics process.
int_rows_deleted	xs:long	int_rows_deleted - Internal Rows Deleted monitor element
int_rows_inserted	xs:long	int_rows_inserted - Internal Rows Inserted monitor element
int_rows_updated	xs:long	int_rows_updated - Internal Rows Updated monitor element
intra_parallel_state	xs:string	intra_parallel_state - Current state of intrapartition parallelism monitor element
invocation_id	xs:nonNegativeInteger	invocation_id - Invocation ID monitor element
is_system_appl	xs:short	is_system_appl - Is System Application monitor element
last_reference_time	xs:dateTime	last_reference_time - Last reference time. Every time a request occurs in this activity, this field is updated.
lob_prefetch_reqs	xs:nonNegativeInteger	lob_prefetch_reqs - LOB prefetcher request count monitor element
lob_prefetch_wait_time	xs:nonNegativeInteger	lob_prefetch_wait_time - LOB prefetch wait time monitor element
local_start_time	xs:dateTime	local_start_time - Local start time.

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
lock_escals	xs:nonNegativeInteger	lock_escals - Number of lock escalations
lock_escals_global	xs:long	lock_escals_global - Number of global lock escalations monitor element
lock_escals_locklist	xs:long	lock_escals_locklist - Number of locklist lock escalations monitor element
lock_escals_maxlocks	xs:long	lock_escals_maxlocks - Number of maxlocks lock escalations monitor element
lock_timeouts	xs:nonNegativeInteger	lock_timeouts - Number of lock timeouts
lock_timeouts_global	xs:long	lock_timeouts_global - Lock timeouts global monitor element
lock_wait_time	xs:nonNegativeInteger	lock_wait_time - Time waited on locks
lock_wait_time_global	xs:long	lock_wait_time_global - Lock wait time global monitor element
lock_waits	xs:nonNegativeInteger	lock_waits - Lock waits
lock_waits_global	xs:long	lock_waits_global - Lock waits global monitor element
log_buffer_wait_time	xs:nonNegativeInteger	log_buffer_wait_time - Log buffer wait time
log_disk_wait_time	xs:nonNegativeInteger	log_disk_wait_time - Log disk wait time
log_disk_waits_total	xs:nonNegativeInteger	log_disk_waits_total - Log disk waits total
member	xs:nonNegativeInteger	member - Database member
nesting_level	xs:nonNegativeInteger	nesting_level - Nesting level monitor element
num_agents	xs:long	num_agents - Number of Agents Working on a Statement monitor element
num_log_buffer_full	xs:nonNegativeInteger	num_log_buffer_full - Number of full log buffers
num_lw_thresh_exceeded	xs:nonNegativeInteger	num_lw_thresh_exceeded - Number of thresholds exceeded
num_routines	xs:int	num_routines - Number of routines
olap_func_overflows	xs:long	olap_func_overflows - OLAP Function Overflows monitor element
package_name	xs:string (128)	package_name - Package name
package_schema	xs:string (128)	package_schema - Package schema
package_version_id	xs:string (128)	package_version_id - Package version
parent_activity_id	xs:nonNegativeInteger	parent_activity_id - Parent activity ID
parent_uow_id	xs:nonNegativeInteger	parent_uow_id - Parent unit of work ID
planid	xs:long	planid - Query plan ID monitor element
pool_col_gbp_indep_pages_found_in_lbp	xs:nonNegativeInteger	pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool
pool_col_gbp_invalid_pages	xs:nonNegativeInteger	pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages
pool_col_gbp_l_reads	xs:nonNegativeInteger	pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads
pool_col_gbp_p_reads	xs:nonNegativeInteger	pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
pool_col_l_reads	xs:nonNegativeInteger	pool_col_l_reads - Buffer pool column-organized logical reads
pool_col_lbp_pages_found	xs:nonNegativeInteger	pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found
pool_col_p_reads	xs:nonNegativeInteger	pool_col_p_reads - Buffer pool column-organized physical reads
pool_col_writes	xs:nonNegativeInteger	pool_col_writes - Buffer pool column-organized writes
pool_data_gbp_indep_pages_found_in_lbp	xs:nonNegativeInteger	pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element
pool_data_gbp_invalid_pages	xs:long	pool_data_gbp_invalid_pages - Group buffer pool invalid data pages monitor element
pool_data_gbp_l_reads	xs:long	pool_data_gbp_l_reads - Group buffer pool data logical reads monitor element
pool_data_gbp_p_reads	xs:long	pool_data_gbp_p_reads - Group buffer pool data physical reads monitor element
pool_data_l_reads	xs:nonNegativeInteger	pool_data_l_reads - Buffer pool data logical reads
pool_data_lbp_pages_found	xs:long	pool_data_lbp_pages_found - Local buffer pool found data pages monitor element
pool_data_p_reads	xs:nonNegativeInteger	pool_data_p_reads - Buffer pool data physical reads
pool_data_writes	xs:nonNegativeInteger	pool_data_writes - Buffer pool data writes
pool_failed_async_col_reqs	xs:nonNegativeInteger	pool_failed_async_col_reqs - Failed column-organized prefetch requests
pool_failed_async_data_reqs	xs:nonNegativeInteger	pool_failed_async_data_reqs - Failed data prefetch requests monitor element
pool_failed_async_index_reqs	xs:nonNegativeInteger	pool_failed_async_index_reqs - Failed index prefetch requests monitor element
pool_failed_async_other_reqs	xs:nonNegativeInteger	pool_failed_async_other_reqs - Failed non-prefetch requests monitor element
pool_failed_async_temp_data_reqs	xs:nonNegativeInteger	pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element
pool_failed_async_temp_index_reqs	xs:nonNegativeInteger	pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element
pool_failed_async_temp_xda_reqs	xs:nonNegativeInteger	pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element
pool_failed_async_temp_col_reqs	xs:nonNegativeInteger	pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests
pool_failed_async_xda_reqs	xs:nonNegativeInteger	pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element
pool_index_gbp_indep_pages_found_in_lbp	xs:nonNegativeInteger	pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element
pool_index_gbp_invalid_pages	xs:long	pool_index_gbp_invalid_pages - Group buffer pool invalid index pages monitor element
pool_index_gbp_l_reads	xs:long	pool_index_gbp_l_reads - Group buffer pool index logical reads monitor element

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
pool_index_gbp_p_reads	xs:long	pool_index_gbp_p_reads - Group buffer pool index physical reads monitor elements
pool_index_l_reads	xs:nonNegativeInteger	pool_index_l_reads - Buffer pool index logical reads
pool_index_lbp_pages_found	xs:long	pool_index_lbp_pages_found - Local buffer pool index pages found monitor element
pool_index_p_reads	xs:nonNegativeInteger	pool_index_p_reads - Buffer pool index physical reads
pool_index_writes	xs:nonNegativeInteger	pool_index_writes - Buffer pool index writes
pool_queued_async_col_pages	xs:nonNegativeInteger	pool_queued_async_col_pages - Column-organized page prefetch requests
pool_queued_async_col_reqs	xs:nonNegativeInteger	pool_queued_async_col_reqs - Column-organized prefetch requests
pool_queued_async_data_pages	xs:nonNegativeInteger	pool_queued_async_data_pages - Data pages prefetch requests monitor element
pool_queued_async_data_reqs	xs:nonNegativeInteger	pool_queued_async_data_reqs - Data prefetch requests monitor element
pool_queued_async_index_pages	xs:nonNegativeInteger	pool_queued_async_index_pages - Index pages prefetch requests monitor element
pool_queued_async_index_reqs	xs:nonNegativeInteger	pool_queued_async_index_reqs - Index prefetch requests monitor element
pool_queued_async_temp_col_pages	xs:nonNegativeInteger	pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests
pool_queued_async_temp_col_reqs	xs:nonNegativeInteger	pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests
pool_queued_async_other_reqs	xs:nonNegativeInteger	pool_queued_async_other_reqs - Non-prefetch requests monitor element
pool_queued_async_temp_data_pages	xs:nonNegativeInteger	pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element
pool_queued_async_temp_data_reqs	xs:nonNegativeInteger	pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element
pool_queued_async_temp_index_pages	xs:nonNegativeInteger	pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element
pool_queued_async_temp_index_reqs	xs:nonNegativeInteger	pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element
pool_queued_async_temp_xda_pages	xs:nonNegativeInteger	pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element
pool_queued_async_temp_xda_reqs	xs:nonNegativeInteger	pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element
pool_queued_async_xda_pages	xs:nonNegativeInteger	pool_queued_async_xda_pages - XDA pages prefetch requests
pool_queued_async_xda_reqs	xs:nonNegativeInteger	pool_queued_async_xda_reqs - XDA prefetch requests monitor element
pool_read_time	xs:nonNegativeInteger	pool_read_time - Total buffer pool physical read time
pool_temp_col_l_reads	xs:nonNegativeInteger	pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
pool_temp_col_p_reads	xs:nonNegativeInteger	pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads
pool_temp_data_l_reads	xs:nonNegativeInteger	pool_temp_data_l_reads - Buffer pool temporary data logical reads
pool_temp_data_p_reads	xs:nonNegativeInteger	pool_temp_data_p_reads - Buffer pool temporary data physical reads
pool_temp_index_l_reads	xs:nonNegativeInteger	pool_temp_index_l_reads - Buffer pool temporary index logical reads
pool_temp_index_p_reads	xs:nonNegativeInteger	pool_temp_index_p_reads - Buffer pool temporary index physical reads
pool_temp_xda_l_reads	xs:nonNegativeInteger	pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads
pool_temp_xda_p_reads	xs:nonNegativeInteger	pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads
pool_write_time	xs:nonNegativeInteger	pool_write_time - Total buffer pool physical write time
pool_xda_gbp_indep_pages_found_in_lbp	xs:nonNegativeInteger	pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element
pool_xda_gbp_invalid_pages	xs:nonNegativeInteger	pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages
pool_xda_gbp_l_reads	xs:nonNegativeInteger	pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests
pool_xda_gbp_p_reads	xs:nonNegativeInteger	pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests
pool_xda_l_reads	xs:nonNegativeInteger	pool_xda_l_reads - Buffer pool XDA data logical reads
pool_xda_lbp_pages_found	xs:nonNegativeInteger	pool_xda_lbp_pages_found - Local buffer pool XDA data pages found
pool_xda_p_reads	xs:nonNegativeInteger	pool_xda_p_reads - Buffer pool XDA data physical reads
pool_xda_writes	xs:nonNegativeInteger	pool_xda_writes - Buffer pool XDA data writes
post_shrthreshold_hash_joins	xs:long	post_shrthreshold_hash_joins - Post threshold hash joins monitor element
post_shrthreshold_sorts	xs:nonNegativeInteger	post_shrthreshold_sorts - Post shared threshold sorts
post_threshold_col_vector_consumers	xs:long	post_threshold_col_vector_consumers - Post threshold columnar_vector consumers
post_threshold_hash_grpbys	xs:long	post_threshold_hash_grpbys - Hash group by threshold post_threshold_hash_grpbys - Hash group by threshold
post_threshold_hash_joins	xs:long	post_threshold_hash_joins - Hash Join Threshold monitor element
post_threshold_olap_funcs	xs:long	post_threshold_olap_funcs - OLAP function threshold monitor element
post_threshold_peas	xs:long	post_threshold_peas - Partial early aggregation threshold monitor element
post_threshold_peds	xs:long	post_threshold_peds - Partial early distincts threshold monitor element
post_threshold_sorts	xs:nonNegativeInteger	post_threshold_sorts - Post threshold sorts

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
prefetch_wait_time	xs:nonNegativeInteger	prefetch_wait_time - Time waited for prefetch
prefetch_waits	xs:nonNegativeInteger	prefetch_waits - Prefetcher wait count monitor element
query_actual_degree	xs:int	query_actual_degree - Actual runtime degree of intrapartition parallelism monitor element
query_card_estimate	xs:long	query_card_estimate - Query Number of Rows Estimate monitor element
query_cost_estimate	xs:integer	query_cost_estimate - Query cost estimate
query_data_tag_list	xs:string(32)	query_data_tag_list - Query data tag list
reclaim_wait_time	xs:long	reclaim_wait_time - Reclaim wait time monitor element
routine_id	xs:nonNegativeInteger	routine_id - Routine ID
rows_deleted	xs:long	rows_deleted - Rows deleted monitor element
rows_inserted	xs:long	rows_inserted - Rows inserted monitor element
rows_modified	xs:nonNegativeInteger	rows_modified - Rows modified
rows_read	xs:nonNegativeInteger	rows_read - Rows read
rows_returned	xs:nonNegativeInteger	rows_returned - Rows returned
rows_updated	xs:long	rows_updated - Rows updated monitor element
section_number	xs:integer	section_number - Section number
semantic_env_id	xs:long	semantic_env_id - Query semantic compilation environment ID monitor element
service_class_id	xs:integer	service_class_id - Service class
service_class_work_class_id	xs:nonNegativeInteger	service_class_work_class_id - Service class work class ID monitor element
service_class_work_action_set_id	xs:nonNegativeInteger	service_class_work_action_set_id - Service class work action set ID monitor element
sort_consumer_heap_top	xs:long	sort_consumer_heap_top - Individual private sort heap consumer high watermark
sort_consumer_shrheap_top	xs:long	sort_consumer_shrheap_top - Individual shared sort heap consumer high watermark
sort_heap_allocated	xs:long	sort_heap_allocated - Total Sort Heap Allocated monitor element
sort_heap_top	xs:long	sort_heap_top - Sort private heap high watermark monitor element
sort_overflows	xs:nonNegativeInteger	sort_overflows - Sort overflows
sort_shrheap_allocated	xs:long	sort_shrheap_allocated - Sort Share Heap Currently Allocated monitor element
sort_shrheap_top	xs:long	sort_shrheap_top - Sort share heap high watermark monitor element
spacemappage_reclaim_wait_time	xs:long	spacemappage_reclaim_wait_time - Space map page reclaim wait time monitor element
sqlrowsread_threshold_id	xs:int	sqlrowsread_threshold_id - SQL rows read threshold ID monitor element
sqlrowsread_threshold_value	xs:long	sqlrowsread_threshold_value - SQL rows read threshold value

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
sqlrowsread_threshold_violated	xs:short (1 = yes, 0 = no)	sqlrowsread_threshold_violated - SQL rows read threshold violated
sqlrowsreadinsc_threshold_id	xs:int	sqlrowsreadinsc_threshold_id - SQL rows read in service threshold ID
sqlrowsreadinsc_threshold_value	xs:long	sqlrowsreadinsc_threshold_value - SQL rows read in service threshold value
sqlrowsreadinsc_threshold_violated	xs:short (1 = yes, 0 = no)	sqlrowsreadinsc_threshold_violated - SQL rows read in service threshold violated
sqlrowsreturned_threshold_id	xs:int	sqlrowsreturned_threshold_id - SQL rows read returned threshold ID
sqlrowsreturned_threshold_value	xs:long	sqlrowsreturned_threshold_value - SQL rows read returned threshold value
sqlrowsreturned_threshold_violated	xs:short (1 = yes, 0 = no)	sqlrowsreturned_threshold_violated - SQL rows read returned threshold violated
sqltempespace_threshold_id	xs:int	sqltempespace_threshold_id - SQL temporary space threshold ID
sqltempespace_threshold_value	xs:long	sqltempespace_threshold_value - SQL temporary space threshold value
sqltempespace_threshold_violated	xs:short (1 = yes, 0 = no)	sqltempespace_threshold_violated - SQL temporary space threshold violated
stmt_exec_time	xs:nonNegativeInteger	stmt_exec_time - Statement execution time
stmt_pkg_cache_id	xs:nonNegativeInteger	stmt_pkgcache_id - Statement package cache identifier
stmt_text	xs:string	stmt_text - SQL statement text. If the activity is dynamic SQL or it is static SQL for which the statement text is available, this field contains the first 1024 characters of the statement text. Otherwise, it contains an empty string.
stmtid	xs:long	stmtid - Query statement ID monitor element
stmtno	xs:int	stmtno - Statement number monitor element
thresh_violations	xs:nonNegativeInteger	thresh_violations - Number of threshold violations
total_act_time	xs:nonNegativeInteger	total_act_time - Total activity time
total_act_wait_time	xs:nonNegativeInteger	total_act_wait_time - Total activity wait time
total_app_section_executions	xs:nonNegativeInteger	total_app_section_executions - Total section executions
total_col_executions	xs:nonNegativeInteger	total_col_executions - Total column-organized executions
total_col_proc_time	xs:nonNegativeInteger	total_col_proc_time - Total column-organized processing time
total_col_synopsis_executions	xs:nonNegativeInteger	total_col_synopsis_executions - Total column-organized synopsis table executions monitor element
total_col_synopsis_proc_time	xs:nonNegativeInteger	total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element
total_col_synopsis_time	xs:nonNegativeInteger	total_col_synopsis_time - Total column-organized synopsis table time monitor element
total_col_time	xs:nonNegativeInteger	total_col_time - Total column-organized time
total_col_vector_consumers	xs:long	total_col_vector_consumers - Total columnar_vector consumers
total_cpu_time	xs:nonNegativeInteger	total_cpu_time - Total CPU time

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
total_disp_run_queue_time	xs:long	total_disp_run_queue_time - Total dispatcher run queue time monitor element in microseconds
total_extended_latch_wait_time	xs:nonNegativeInteger	total_extended_latch_wait_time - Total extended latch wait time monitor element
total_extended_latch_waits	xs:nonNegativeInteger	total_extended_latch_waits - Total extended latch waits monitor element
total_hash_grpbys	xs:long	total_hash_grpbys - Total hash group by operations
total_hash_joins	xs:long	total_hash_joins - Total Hash Joins monitor element
total_hash_loops	xs:long	total_hash_loops - Total Hash Loops monitor element
total_index_build_proc_time	xs:long	total_index_build_proc_time - Total non-wait time spent building indexes due to index creation
total_index_build_time	xs:long	total_index_build_time - Total time spent building indexes due to index creation
total_indexes_built	xs:long	total_indexes_built - Total number of indexes built
total_olap_funcs	xs:long	total_olap_funcs - Total OLAP Functions monitor element
total_peas	xs:long	total_peas - Total partial early aggregations monitor element
total_peds	xs:long	total_peds - Total partial early distincts monitor element
total_routine_invocations	xs:nonNegativeInteger	total_routine_invocations - Total routine invocations
total_routine_non_sect_proc_time	xs:nonNegativeInteger	total_routine_non_sect_proc_time - Non-section processing time
total_routine_non_sect_time	xs:nonNegativeInteger	total_routine_non_sect_time - Non-section routine execution time
total_routine_time	xs:nonNegativeInteger	total_routine_time - Total routine time
total_routine_user_code_proc_time	xs:nonNegativeInteger	total_routine_user_code_proc_time - Total routine user code processing time
total_routine_user_code_time	xs:nonNegativeInteger	total_routine_user_code_time - Total routine user code time
total_section_proc_time	xs:nonNegativeInteger	total_section_proc_time - Total section processing time
total_section_sort_proc_time	xs:nonNegativeInteger	total_section_sort_proc_time - Total section sort processing time
total_section_sort_time	xs:nonNegativeInteger	total_section_sort_time - Total section sort time.
total_section_sorts	xs:nonNegativeInteger	total_section_sorts - Total section sorts.
total_section_time	xs:nonNegativeInteger	total_section_time - Total section time
total_sorts	xs:nonNegativeInteger	total_sorts - Total Sorts
total_sync_runstats	xs:nonNegativeInteger	total_sync_runstats - Total synchronous RUNSTATS activities monitor element
tq_sort_heap_rejections	xs:long	tq_sort_heap_rejections - Table queue sort heap rejections monitor element
tq_sort_heap_requests	xs:long	tq_sort_heap_requests - Table queue sort heap requests monitor element
tq_tot_send_spills	xs:nonNegativeInteger	tq_tot_send_spills - Total number of tablequeue buffers overflowed
uow_id	xs:nonNegativeInteger	uow_id - Unit of work ID

Table 109. Detailed metrics returned for MON\_GET\_ACTIVITY\_DETAILS (continued)

Element name	Data type	Description or corresponding monitor element
utility_id	xs:nonNegativeInteger	utility_id - Utility ID
utility_invocation_id	xs:hexBinary	utility_invocation_id - Utility invocation ID monitor element
wl_work_action_set_id	xs:nonNegativeInteger	wl_work_action_set_id - Workload work action set identifier monitor element
wl_work_class_id	xs:nonNegativeInteger	wl_work_class_id - Workload work class identifier
wlm_queue_assignments_total	xs:nonNegativeInteger	wlm_queue_assignments_total - Workload manager total queue assignments
wlm_queue_time_total	xs:nonNegativeInteger	wlm_queue_time_total - Workload manager total queue time

## Usage notes

The MON\_GET\_ACTIVITY\_DETAILS function provides maximum flexibility for formatting output because it returns detailed information for a single activity as an XML document. The XML output includes both descriptive information (for example, statement text) and metrics. The output can be parsed directly by an XML parser, or it can be converted to relational format by the XMLTABLE function as shown in the example.

The metrics reported through this function (for example, CPU usage) are rolled up to the activity periodically during the lifetime of the activity. Therefore, the values reported by this table function reflect the current state of the system at the time of the most recent rollup.

Activity metrics are controlled through the COLLECT ACTIVITY METRICS clause on workloads, or the mon\_act\_metrics database configuration parameter at the database level. Metrics are collected if the connection that submits the activity is associated with a workload or database for which activity metrics are enabled. If activity metrics are not collected for an activity, all metrics are reported as 0.

The MON\_GET\_ACTIVITY\_DETAILS table function returns one row of data for each member on which the activity exists. No aggregation across members is performed for the metrics. However, aggregation can be achieved through SQL queries.

The schema for the XML document that is returned in the DETAILS column is available in the file sqllib/misc/DB2MonRoutines.xsd. Further details can be found in the file sqllib/misc/DB2MonCommon.xsd.

## Example

**Investigate a long running query to determine if it is spending its time executing or waiting (for example, blocked on locks or I/O).**

Note: The following queries can be combined into one statement and are shown in 2 steps for reasons of clarity. Also, if you want to retrieve the complete text, you could use the executable ID to obtain the statement text from the MON\_GET\_PKG\_CACHE\_STMT table function.

1. First use the WLM\_GET\_WORKLOAD\_OCCURRENCE\_ACTIVITIES table function to list activities and their start times.

```

SELECT application_handle,
 activity_id,
 uow_id,
 local_start_time
FROM TABLE(
 WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES(
 cast(NULL as bigint), -1)
) AS T

```

The following is an example of output from this query.

APPLICATION_HANDLE	ACTIVITY_ID	UOW_ID	LOCAL_START_TIME
7	1	2	2008-06-10-10.06.55.675668
16	1	7	2008-06-10-10.08.38.613610

2 record(s) selected.

2. Then use the MON\_GET\_ACTIVITY\_DETAILS table function to view the percentage of time that the activity has spent waiting.

```

SELECT actmetrics.application_handle,
 actmetrics.activity_id,
 actmetrics.uow_id,
 varchar(actmetrics.stmt_text, 50) as stmt_text,
 actmetrics.total_act_time,
 actmetrics.total_act_wait_time,
 CASE WHEN actmetrics.total_act_time > 0
 THEN DEC(
 FLOAT(actmetrics.total_act_wait_time) /
 FLOAT(actmetrics.total_act_time)) * 100, 5, 2)
 ELSE NULL
 END AS PERCENTAGE_WAIT_TIME
FROM TABLE(MON_GET_ACTIVITY_DETAILS(7, 2, 1, -2)) AS ACTDETAILS,
XMLTABLE (XMLNAMESPACES(DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'),
'$actmetrics/db2_activity_details'
PASSING XMLPARSE(DOCUMENT ACTDETAILS.DETAILS) as "actmetrics"
COLUMNS "APPLICATION_HANDLE" INTEGER PATH 'application_handle',
"ACTIVITY_ID" INTEGER PATH 'activity_id',
"UOW_ID" INTEGER PATH 'uow_id',
"STMT_TEXT" VARCHAR(1024) PATH 'stmt_text',
"TOTAL_ACT_TIME" INTEGER PATH 'activity_metrics/total_act_time',
"TOTAL_ACT_WAIT_TIME" INTEGER PATH 'activity_metrics/total_act_wait_time'
) AS ACTMETRICS;

```

The following is an example of output from this query.

APPLICATION_HANDLE	ACTIVITY_ID	UOW_ID	...
7	1	2	...

1 record(s) selected.

Output for query (continued).

... STMT_TEXT	...
... select * from syscat.tables optimize for 1 row	...

Output for query (continued).

... TOTAL_ACT_TIME	TOTAL_ACT_WAIT_TIME	PERCENTAGE_WAIT_TIME	...
459	0	0.00	...

**Use the MON\_GET\_ACTIVITY\_DETAILS table function to create a query that captures information about all the activities currently running on a system.**

- Example 1: Run the following command using the command line processor (CLP)

```

WITH A1 AS
 (SELECT * FROM TABLE(wlm_get_workload_occurrence_activities(null, -1))
 WHERE activity_id > 0)
SELECT A1.application_handle,
 A1.activity_id,
 A1.uow_id,
 total_act_time,
 total_act_wait_time,
 varchar(actmetrics.stmt_text, 50) AS stmt_text FROM A1,
TABLE(MON_GET_ACTIVITY_DETAILS(A1.application_handle, A1.uow_id,A1.activity_id, -1))
 AS ACTDETAILS,
XMLTABLE (XMLNAMESPACES(DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'),
 '$actmetrics/db2_activity_details'
 PASSING XMLPARSE(DOCUMENT ACTDETAILS.DETAILS) AS "actmetrics"
 COLUMNS "STMT_TEXT" VARCHAR(1024) PATH 'stmt_text',
 "TOTAL_ACT_TIME" INTEGER PATH 'activity_metrics/total_act_time',
 "TOTAL_ACT_WAIT_TIME" INTEGER PATH 'activity_metrics/total_act_wait_time')
 AS ACTMETRICS

```

The following is an example of output from this query:

```

APP...HANDLE A..._ID UOW_ID T...ACT_TIME T...WAIT_TIME

15 1 5 16 5
15 1 3 17 5
7 1 49 0 0
SQL0445W Value "with A1 as (select * from table(wlm_get_workload
3 record(s) selected with 1 warning messages printed.

```

The following sample continues output from this query:

```

... STMT_TEXT
... -----
... select name from sysibm.systables
... select * from sysibm.systables
... with A1 as (select * from table(wlm_get_workload_o
occurrence" has been truncated. SQLSTATE=01004

3 record(s) selected with 1 warning messages printed.

```

## MON\_GET\_AGENT table function - List agents, fenced mode processes, and system entities for the database

The MON\_GET\_AGENT function returns a list of all agents, fenced mode processes (db2fmp processes), and system entities for the database. The list can be filtered to show information for a specified member, service class, or application.

The system entities are non-agent threads and processes, such as page cleaners and prefetchers.

Refer to Table 110 on page 552 for a complete list of information that can be returned.

### Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

### Default PUBLIC privilege

None

## Syntax

```
►► MON_GET_AGENT (—service_superclass_name—, —————►
► service_subclass_name—, —application_handle—, —member—) —————►◄
```

The schema is SYSPROC.

## Routine parameters

### *service\_superclass\_name*

An input argument of type VARCHAR(128) that specifies the name of a service superclass in the currently connected database. If the argument is null or an empty string, data is retrieved for all the superclasses in the database.

### *service\_subclass\_name*

An input argument of type VARCHAR(128) that refers to a specific subclass within a superclass. If the argument is null or an empty string, data is retrieved for all the subclasses in the database.

### *application\_handle*

An input argument of type BIGINT that specifies the application handle for which agent information is to be returned. If the argument is null, data is retrieved for all applications in the database. An application handle of 0 returns the system entities only.

### *member*

An input argument of type INTEGER that specifies the member number in the same instance as the currently connected database. Specify -1 for the current database member, or -2 for all active database members. If a null value is specified, -1 is set implicitly.

## Information returned

Table 110. Information returned by the MON\_GET\_AGENT routine

Column name	Data type	Description
SERVICE_SUPERCLASS_NAME	VARCHAR (128)	service_superclass_name - Service superclass name monitor element
SERVICE_SUBCLASS_NAME	VARCHAR (128)	service_subclass_name - Service subclass name monitor element
APPLICATION_HANDLE	BIGINT	application_handle - Application handle monitor element
DBPARTITIONNUM	SMALLINT	dbpartitionnum - Database partition number monitor element
ENTITY	VARCHAR (32)	One of the following values: <ul style="list-style-type: none"><li>• If the type of entity is an agent, the value is db2agent.</li><li>• If the type of entity is a fenced mode process, the value is db2fmp (<i>pid</i>) where <i>pid</i> is the process ID of the fenced mode process.</li><li>• Otherwise, the value is the name of the system entity.</li></ul>
WORKLOAD_NAME	VARCHAR (128)	workload_name - Workload name monitor element
WORKLOAD_OCCURRENCE_ID	INTEGER	workload_occurrence_id - Workload occurrence identifier monitor element
UOW_ID	INTEGER	uow_id - Unit of work ID monitor element

Table 110. Information returned by the MON\_GET\_AGENT routine (continued)

Column name	Data type	Description
ACTIVITY_ID	INTEGER	activity_id - Activity ID monitor element
PARENT_UOW_ID	INTEGER	parent_uow_id - Parent unit of work ID monitor element
PARENT_ACTIVITY_ID	INTEGER	parent_activity_id - Parent activity ID monitor element
AGENT_TID	BIGINT	agent_tid - Agent thread ID monitor element
AGENT_TYPE	VARCHAR (32)	<p>Agent type. The agent types are as follows:</p> <ul style="list-style-type: none"> <li>• COORDINATOR</li> <li>• OTHER</li> <li>• PDBSUBAGENT</li> <li>• SMPSUBAGENT</li> </ul> <p>If the value is COORDINATOR, the agent ID might change in concentrator environments.</p> <p>The SMPSUBAGENT type is returned for agents that are processing column-organized tables.</p>
SMP_COORDINATOR	INTEGER	Indication of whether the agent is an SMP coordinator: 1 for yes and 0 for no.
AGENT_SUBTYPE	VARCHAR (32)	<p>Agent subtype. The possible subtypes are as follows:</p> <ul style="list-style-type: none"> <li>• DSS</li> <li>• OTHER</li> <li>• RPC</li> <li>• SMP</li> <li>• COLUMNAR</li> </ul>
AGENT_STATE	VARCHAR (32)	<p>Indication of whether an agent is associated or active. The possible values are:</p> <ul style="list-style-type: none"> <li>• ASSOCIATED</li> <li>• ACTIVE</li> </ul>
EVENT_TYPE	VARCHAR (32)	<p>Type of event last processed by this agent. The possible values are as follows:</p> <ul style="list-style-type: none"> <li>• ACQUIRE</li> <li>• PROCESS</li> <li>• WAIT</li> </ul> <p>See Table 111 on page 557 for more information about possible values for this column.</p>

Table 110. Information returned by the `MON_GET_AGENT` routine (continued)

Column name	Data type	Description
EVENT_OBJECT	VARCHAR (32)	<p>Object of the event last processed by this agent. The possible values are as follows:</p> <ul style="list-style-type: none"> <li>• COMPRESSION_DICTIONARY_BUILD</li> <li>• IMPLICIT_REBIND</li> <li>• INDEX_RECREATE</li> <li>• LATCH</li> <li>• LOCK</li> <li>• LOCK_ESCALATION</li> <li>• QP_QUEUE</li> <li>• REMOTE_REQUEST</li> <li>• REQUEST</li> <li>• ROUTINE</li> <li>• WLM_QUEUE</li> <li>• TQ_SEND</li> <li>• TQ_RECEIVE</li> </ul> <p>See Table 111 on page 557 for more information about possible values for this column.</p>
EVENT_STATE	VARCHAR (32)	<p>State of the event last processed by this agent. The possible values are as follows:</p> <ul style="list-style-type: none"> <li>• EXECUTING</li> <li>• IDLE</li> </ul> <p>See Table 111 on page 557 for more information about possible values for this column.</p>
REQUEST_ID	VARCHAR (64)	<p>Request ID. This value is unique only in combination with the value of <i>application_handle</i>. You can use this combination to distinguish between one request that is taking a long time and multiple requests; for example, to distinguish between one long fetch and multiple fetches.</p>

Table 110. Information returned by the MON\_GET\_AGENT routine (continued)

Column name	Data type	Description
REQUEST_TYPE	VARCHAR (32)	<p>Type of request. The possible values are as follows:</p> <ul style="list-style-type: none"> <li>• For coordinator agents: <ul style="list-style-type: none"> <li>– CLOSE</li> <li>– COMMIT</li> <li>– COMPILE</li> <li>– DESCRIBE</li> <li>– EXCSQLSET</li> <li>– EXECIMMD</li> <li>– EXECUTE</li> <li>– FETCH</li> <li>– INTERNAL <i>number</i>, where <i>number</i> is the value of the internal constant</li> <li>– OPEN</li> <li>– PREPARE</li> <li>– REBIND</li> <li>– REDISTRIBUTE</li> <li>– REORG</li> <li>– ROLLBACK</li> <li>– RUNSTATS</li> </ul> </li> <li>• For subagents with an AGENT_SUBTYPE column value of DSS, SMP, or COLUMNAR: <ul style="list-style-type: none"> <li>– If the subsection number is nonzero, the subsection number in the form SUBSECTION:<i>subsection_number</i>; otherwise, NULL.</li> </ul> </li> </ul>

Table 110. Information returned by the MON\_GET\_AGENT routine (continued)

Column name	Data type	Description
REQUEST_TYPE (continued)	VARCHAR (32)	<ul style="list-style-type: none"> <li>For subagents with an AGENT_SUBTYPE of RPC: <ul style="list-style-type: none"> <li>ABP</li> <li>CATALOG</li> <li>INTERNAL</li> <li>REORG</li> <li>RUNSTATS</li> <li>WLM</li> </ul> </li> <li>For subagents with a SUBTYPE of OTHER: <ul style="list-style-type: none"> <li>ABP</li> <li>APP_RBSVPT</li> <li>APP_RELSVPT</li> <li>BACKUP</li> <li>CLOSE</li> <li>EXTERNAL_RBSVPT</li> <li>EVMON</li> <li>FORCE</li> <li>FORCE_ALL</li> <li>INTERNAL <i>number</i>, where <i>number</i> is the value of the internal constant</li> <li>INTERRUPT</li> <li>NOOP (if there is no request)</li> <li>QP</li> <li>REDISTRIBUTE</li> <li>STMT_RBSVPT</li> <li>STOP_USING</li> <li>UPDATE_DBM_CFG</li> <li>WLM</li> </ul> </li> </ul>
NESTING_LEVEL	INTEGER	nesting_level - Nesting level monitor element
INVOCATION_ID	INTEGER	invocation_id - Invocation ID monitor element
ROUTINE_ID	INTEGER	routine_id - Routine ID monitor element
EVENT_OBJECT_NAME	VARCHAR (1024)	<p>Event object name.</p> <ul style="list-style-type: none"> <li>If the value of EVENT_OBJECT is LOCK, the value of this column is the name of the lock that the agent is waiting on.</li> <li>If the value of EVENT_OBJECT is WLM_QUEUE, the value of the column is the name of the WLM threshold that the agent is queued on.</li> <li>If the value of EVENT_OBJECT is TQ_SEND or TQ_RECEIVE the value of the column represents the table queue ID number.</li> <li>Otherwise, the value is NULL.</li> </ul>
APPLICATION_NAME	VARCHAR (128)	appl_name - Application name
APPLICATION_ID	VARCHAR (128)	appl_id - Application ID
CLIENT_PID	BIGINT	client_pid - Client process ID

Table 110. Information returned by the MON\_GET\_AGENT routine (continued)

Column name	Data type	Description
SESSION_AUTH_ID	VARCHAR (128)	session_auth_id - Session authorization ID
REQUEST_START_TIME	TIMESTAMP	Time that the agent started processing the request on which it is currently working
AGENT_STATE_LAST_UPDATE_TIME	TIMESTAMP	The last time that the event, being processed by the agent, was changed. The event currently processed by the agent is identified by the EVENT_TYPE, EVENT_OBJECT, and EVENT_STATE columns.
EXECUTABLE_ID	VARCHAR (32) FOR BIT DATA	executable_id - Executable ID monitor element
MEMBER	SMALLINT	member - Database member monitor element
UTILITY_INVOCATION_ID	VARCHAR (32) FOR BIT DATA	utility_invocation_id - Utility invocation ID
IS_SYSTEM_APPL	SMALLINT	is_system_appl - Is System Application monitor element
SUBSECTION_NUMBER	BIGINT	Displays subsection ID if an agent (any agent type) is processing a subsection; otherwise NULL.
EVENT_OBJECT_DETAILS	VARCHAR (1024)	Displays detailed information about the event wait object. Basic information about the event wait object is displayed in the EVENT_OBJECT_NAME field.

**Note:** The possible combinations of EVENT\_STATE, EVENT\_TYPE, EVENT\_OBJECT, and EVENT\_OBJECT\_NAME column values are listed in the following table.

Table 111. Possible combinations for EVENT\_STATE, EVENT\_TYPE, EVENT\_OBJECT, and EVENT\_OBJECT\_NAME column values

Event description	EVENT_STATE value	EVENT_TYPE value	EVENT_OBJECT value	EVENT_OBJECT_NAME value	EVENT_OBJECT_DETAILS value
Acquire lock	IDLE	ACQUIRE	LOCK	Lock name	NULL
Escalate lock	EXECUTING	PROCESS	LOCK_ESCALATION	NULL	NULL
Process request	EXECUTING	PROCESS	REQUEST	NULL	NULL
Wait for a new request	IDLE	WAIT	REQUEST	NULL	NULL
Wait for a request to be processed at a remote member	IDLE	WAIT	REMOTE_REQUEST	NULL	NULL
Wait on a WLM threshold queue	IDLE	WAIT	WLM_QUEUE	Threshold name	NULL
Process a routine	EXECUTING	PROCESS	ROUTINE	NULL	NULL
Re-create an index	EXECUTING	PROCESS	INDEX_RECREATE	NULL	NULL
Build compression dictionary	EXECUTING	PROCESS	COMP_DICT_BLD	NULL	NULL
Implicit rebind	EXECUTING	PROCESS	IMPLICIT_REBIND	NULL	NULL

Table 111. Possible combinations for EVENT\_STATE, EVENT\_TYPE, EVENT\_OBJECT, and EVENT\_OBJECT\_NAME column values (continued)

Event description	EVENT_STATE value	EVENT_TYPE value	EVENT_OBJECT value	EVENT_OBJECT_NAME value	EVENT_OBJECT_DETAILS value
Sending data on table queue	IDLE	WAIT	TQ_SEND	Table queue number	Member number that is waiting for a table queue, or -1 for any member.
Receiving data on table queue	IDLE	WAIT	TQ_RECEIVE	Table queue number	Member number that is waiting for a table queue, or -1 for any member.
Latch wait	IDLE	WAIT	LATCH	Latch name	NULL

## Usage notes

The parameters are, in effect, ANDed together. That is, if you specify conflicting input parameters, such as a service superclass SUP\_A and a subclass SUB\_B such that SUB\_B is not a subclass of SUP\_A, no rows are returned.

## Examples

- The following query returns a list of agents that are associated with application handle 1 for all database members. You can determine the application handle by using the **LIST APPLICATIONS** command or the MON\_GET\_CONNECTION table function.

```
SELECT SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHANDLE,
 SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
 SUBSTR(CHAR(AGENT_TID),1,9) AS AGENT_TID,
 SUBSTR(AGENT_TYPE,1,11) AS AGENTTYPE,
 SUBSTR(AGENT_STATE,1,10) AS AGENTSTATE,
 SUBSTR(REQUEST_TYPE,1,12) AS REQTYPE,
 SUBSTR(CHAR(UOW_ID),1,6) AS UOW_ID,
 SUBSTR(CHAR(ACTIVITY_ID),1,6) AS ACT_ID
FROM TABLE(MON_GET_AGENT(CAST(NULL AS VARCHAR(128)),
 CAST(NULL AS VARCHAR(128)), 1, -2)) AS SCDETAILS
ORDER BY APPHANDLE, MEMB, AGENT_TID
```

Sample output is as follows:

APPHANDLE	MEMB	AGENT_TID	AGENTTYPE	AGENTSTATE	REQTYPE	UOW_ID	ACT_ID
1	0	3	COORDINATOR	ACTIVE	FETCH	1	5
1	0	4	SUBAGENT	ACTIVE	SUBSECTION:1	1	5
1	1	2	SUBAGENT	ACTIVE	SUBSECTION:2	1	5

The output shows a coordinator agent and a subagent on member 0 and a subagent on member 1 operating on behalf of an activity with UOW ID 1 and activity ID 5. The AGENTTYPE column with a value of COORDINATOR has a value of FETCH for the REQTYPE column (which indicates the main or initial request type). This means that the type of request is a fetch request for the coordinator agent.

- The following query determines which lock an agent is waiting on:
- ```
select event_object, event_type, event_state, varchar(event_object_name, 30)
       as event_object_name
from table(MON_GET_AGENT('',' ',cast(NULL as bigint), -1)) as t
```

Sample output is as follows:

However, you cannot use the snapshot interface determine the state of each agent that is working on a particular subsection. Some agents might be waiting on different table queues and partitions.

The following query can be used to gather information on the state of various agents involved. This query assumes that you know the application number (for example, 349, in this case):

```
SELECT SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHANDLE,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       SUBSTR(CHAR(AGENT_TID),1,20) AS AGENT_TID,
       SUBSTR(AGENT_SUBTYPE,1,10) AS AGENT_SUBTYPE,
       SUBSTR(AGENT_TYPE,1,11) AS AGENTTYPE,
       SUBSTR(AGENT_STATE,1,10) AS AGENTSTATE,
       SUBSTR(EVENT_STATE,1,10) AS EVENT_STATE,
       SUBSTR(EVENT_TYPE,1,10) AS EVENT_TYPE,
       SUBSTR(EVENT_OBJECT,1,10) AS EVENT_OBJECT,
       SUBSTR(REQUEST_TYPE,1,12) AS REQTYPE,
       SUBSTR(EVENT_OBJECT_NAME,1,10) AS TQ_ID,
       SUBSTR(CHAR(SUBSECTION_NUMBER),1,4) AS SUB_ID,
       SUBSTR(EVENT_OBJECT_DETAILS,1,10) AS TQ_WAIT_MEM
FROM TABLE(MON_GET_AGENT(CAST(NULL AS VARCHAR(128)), CAST(NULL AS
VARCHAR(128)), 349, -2))
AS SCDETAILS
ORDER BY APPHANDLE, MEMB, AGENT_TID
```

Note: EVENT_OBJECT_NAME is displayed as TQ_ID, SUBSECTION_NUMBER is displayed as SUB_ID, and EVENT_OBJECT_DETAILS is displayed as TQ_WAIT_MEM. TQ_WAIT_MEM represents the member that you are waiting for, where -1 means 'any'.

The following is an example of output from this query.

| APPHANDLE | MEMB | AGENT_TID | AGENT_SUBTYPE | AGENTTYPE | AGENTSTATE | ... |
|-----------|------|-----------|---------------|-------------|------------|-----|
| 349 | 0 | 250 | - | COORDINATOR | ACTIVE | ... |
| 349 | 0 | 338 | COLUMNAR | SMPSUBAGENT | ACTIVE | ... |
| 349 | 0 | 343 | COLUMNAR | SMPSUBAGENT | ACTIVE | ... |
| 349 | 0 | 335 | SMP | SMPSUBAGENT | ACTIVE | ... |
| 349 | 0 | 336 | SMP | SMPSUBAGENT | ACTIVE | ... |
| 349 | 1 | 277 | COLUMNAR | SMPSUBAGENT | ACTIVE | ... |
| 349 | 2 | 278 | COLUMNAR | SMPSUBAGENT | ACTIVE | ... |

Output for query (continued).

| ...EVENT_STATE | EVENT_TYPE | EVENT_OBJECT | REQTYPE | TQ_ID | SUB_ID | TQ_WAIT_MEM |
|----------------|------------|--------------|--------------|-------|--------|-------------|
| ...IDLE | WAIT | TQ_RECEIVE | OPEN | 1 | 0 | -1 |
| ...IDLE | WAIT | TQ_SEND | SUBSECTION:2 | 3 | 2 | -1 |
| ...IDLE | WAIT | TQ_RECEIVE | - | 2 | 0 | -1 |
| ...IDLE | WAIT | REQUEST | - | - | 0 | - |
| ...EXECUTING | PROCESS | REQUEST | - | - | 0 | - |
| ...IDLE | WAIT | TQ_SEND | SUBSECTION:2 | 3 | 2 | -1 |
| ...IDLE | WAIT | TQ_SEND | SUBSECTION:2 | 3 | 2 | -1 |

In this example, an agent with an AGENT_TID value of 338 represents partition 0; a value of 277 represents partition 1; a value of 278 represents partition 2. All three of these agents are waiting to send to TQ_ID 3 on all partitions. Sending to all partitions is known as a broadcast. In this case, this broadcast occurs in subsection ID 2. The agent with an ID of 250 on partition 0 is processing subsection 0 and is waiting to receive on TQ_ID 1 from any partition. The agent with an ID of 343 is also waiting to receive on subsection 0 but on TQ_ID 2 from any partition. The agent with IDs 335 and 336 are not waiting on any TQ.

MON_GET_APPL_LOCKWAIT - Get information about locks for which an application is waiting

The MON_GET_APPL_LOCKWAIT table function returns information about all locks that each application's agents (that are connected to the current database) are waiting to acquire.

To get information about locks, use the MON_GET_APPL_LOCKWAIT, MON_FORMAT_LOCK_NAME, and MON_GET_LOCKS, table functions instead of the SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function, and the SNAPLOCK administrative view and SNAP_GET_LOCK table function, which are deprecated in Fixpack 1 of Version 9.7.

►► MON_GET_APPL_LOCKWAIT ((*application_handle* , *member* , *system_appls*)) ►►

The schema is SYSPROC.

Routine parameters

application_handle

An optional input parameter of type BIGINT that specifies a valid application handle in the same database as the one to which you are currently connected. If the argument is null, locks are retrieved for all applications that are currently waiting for locks to be acquired.

member

An input parameter of type INTEGER that specifies a valid member in the same instance as the currently connected database. Specify -1 for the current database member, or -2 for all active members. If the NULL value is specified, -1 is set.

system_appls

An input parameter of type SMALLINT that specifies whether information for system applications is to be returned. The following values are valid:

- 0 or NULL: System application information is not returned. NULL is the default if a value for *system_appls* is not provided.
- 1: User and system application information is returned.

Authorization

One of the following authorities or privilege is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Information returned

The columns that are returned provide information in the following areas:

- The following columns represent details about the lock that the application is currently waiting to acquire:

LOCK_WAIT_START_TIME, LOCK_NAME, LOCK_OBJECT_TYPE,
 LOCK_MODE, LOCK_CURRENT_MODE, LOCK_MODE_REQUESTED,
 LOCK_STATUS, LOCK_ESCALATION, LOCK_ATTRIBUTES, LOCK_RRIID,
 LOCK_COUNT, TBSP_ID, TAB_FILE_ID, SUBSECTION_NUMBER.

- The following columns represent details about the application that is waiting to acquire this lock.

REQ_APPLICATION_HANDLE, REQ_AGENT_TID, REQ_MEMBER,
 REQ_EXECUTABLE_ID

- The following columns represent details about the application that is currently holding the lock.

HLD_APPLICATION_HANDLE, HLD_MEMBER, ADDITIONAL_DETAILS

Table 112. Information returned by the MON_GET_APPL_LOCKWAIT table function

| Column name | Data type | Description or monitor element |
|----------------------|----------------------|--|
| LOCK_WAIT_START_TIME | TIMESTAMP | lock_wait_start_time - Lock Wait Start Timestamp |
| LOCK_NAME | VARCHAR(32) | lock_name - Lock name

The internal name can be formatted using the MON_FORMAT_LOCK_NAME table function to obtain details regarding the lock. For example, the table and table space that the lock references can be found, if this is a table lock. |
| LOCK_OBJECT_TYPE_ID | CHAR(1) FOR BIT DATA | Reserved for future use |
| LOCK_OBJECT_TYPE | VARCHAR(32) | lock_object_type - Lock object type waited on

For possible values, see “lock_object_type - Lock object type waited on monitor element” |
| LOCK_MODE | VARCHAR(3) | lock_mode - Lock mode

If the application holding this lock cannot be found, a value of NULL is returned.

For a global lockwait, this value is NULL. |
| LOCK_CURRENT_MODE | VARCHAR(3) | lock_current_mode - Original Lock Mode Before Conversion

If no conversion took place, then a value of NULL is returned. |
| LOCK_MODE_REQUESTED | VARCHAR(3) | lock_mode_requested - Lock mode requested |
| LOCK_STATUS | CHAR(1) | lock_status - Lock status |
| LOCK_ESCALATION | CHAR(1) | lock_escalation - Lock escalation |
| LOCK_ATTRIBUTES | CHAR(16) | lock_attributes - Lock attributes |
| LOCK_RRIID | BIGINT | lock_count - Lock count monitor element |

Table 112. Information returned by the MON_GET_APPL_LOCKWAIT table function (continued)

| Column name | Data type | Description or monitor element |
|------------------------|---------------------------|--|
| LOCK_COUNT | BIGINT | lock_count - Lock count monitor element |
| TBSP_ID | BIGINT | tablespace_id - Table space ID |
| TAB_FILE_ID | BIGINT | table_file_id - Table file ID |
| SUBSECTION_NUMBER | BIGINT | ss_number - Subsection Number

If the subsection number is not available, then a value of NULL is returned. |
| REQ_APPLICATION_HANDLE | BIGINT | req_application_handle - Requesting application handle |
| REQ_AGENT_TID | BIGINT | req_agent_tid - Requesting agent TID |
| REQ_MEMBER | SMALLINT | req_member - Requesting member |
| REQ_EXECUTABLE_ID | VARCHAR (32) FOR BIT DATA | req_executable_id - Requesting executable ID |
| HLD_APPLICATION_HANDLE | BIGINT | hld_application_handle - Holding application handle

The value 0 indicates that the application no longer exists. The value can appear when incomplete transactions are on the database after a system crash. It means that the lock is held by either a transaction that is being rolled back or by an indoubt transaction that must be resolved.

If the application holding this lock is unknown or cannot be found then a value of NULL is returned.

For a global lockwait, this value is NULL. |
| HLD_MEMBER | SMALLINT | hld_member - Holding member |
| IS_SYSTEM_APPL | SMALLINT | is_system_appl - Is System Application |

Example

In this sample scenario, the MON_GET_APPL_LOCKWAIT table function is used to investigate a hung application for the session authorization ID USER1.

1. Use the MON_GET_CONNECTION table function to look up the application handle for all connections with the SESSION_USER value of USER1:

```
SELECT COORD_PARTITION_NUM, APPLICATION_HANDLE
FROM TABLE(MON_GET_CONNECTION(NULL,-2))
WHERE SESSION_USER = 'USER1'
```

This query returns the following output:

| COORD_PARTITION_NUM | APPLICATION_HANDLE |
|---------------------|--------------------|
| 2 | 131130 |

1 record(s) selected.

- Use the MON_GET_AGENT table function to obtain current information about all agents working for this connection, on all database partitions:

```
SELECT SUBSTR(CHAR(DBPARTITIONNUM),1,3) AS DBPART,
       SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APP_ID,
       SUBSTR(CHAR(WORKLOAD_OCCURRENCE_ID),1,7) AS WLO_ID,
       SUBSTR(CHAR(AGENT_TID),1,7) AS AGENT_ID,
       SUBSTR(CHAR(AGENT_TYPE),1,12) AS AGENT_TYPE,
       SUBSTR(AGENT_STATE,1, 8) AS STATE,
       SUBSTR(EVENT_TYPE,1, 8) AS EV_TYPE,
       SUBSTR(EVENT_OBJECT,1,12) AS EV_OBJECT
FROM TABLE(MON_GET_AGENT('',' ',131130,-2))
ORDER BY AGENT_TYPE, DBPART
```

This query returns the following output:

| DBPART | APP_ID | WLO_ID | AGENT_ID | AGENT_TYPE | STATE | EV_TYPE | EV_OBJECT |
|--------|--------|--------|----------|-------------|--------|---------|-----------|
| 2 | 131130 | 1 | 3110 | COORDINATOR | ACTIVE | WAIT | REQUEST |
| 0 | 131130 | 1 | 7054 | PDBSUBAGENT | ACTIVE | ACQUIRE | LOCK |
| 1 | 131130 | 1 | 5709 | PDBSUBAGENT | ACTIVE | ACQUIRE | LOCK |
| 2 | 131130 | 1 | 5960 | PDBSUBAGENT | ACTIVE | ACQUIRE | LOCK |

4 record(s) selected.

An event of type ACQUIRE on an event object of type LOCK indicates a lock wait scenario. Now you can investigate which object is being waited for and which process is holding the lock on it.

- To determine all locks that the application is waiting for, call the MON_GET_APPL_LOCKWAIT table function with application handle 131130 and member -2 as input parameters.

```
SELECT lock_name,
       hld_member AS member,
       hld_agent_tid as TID,
       hld_application_handle AS HLD_APP FROM
TABLE (MON_GET_APPL_LOCKWAIT(131130, -2))
```

This query returns the following output:

| LOCK_NAME | MEMBER | TID | HLD_APP |
|----------------------------|--------|------|---------|
| 00030005000000000280000452 | 0 | 1234 | 65564 |
| 00030005000000000280000452 | 1 | 5478 | 65564 |
| 00030005000000000280000452 | 2 | 4678 | 65564 |

3 record(s) selected.

- Call the MON_GET_CONNECTION table function to find out more about the application that is holding the lock (this application has an application handle of 65564).

```
SELECT SYSTEM_AUTH_ID, APPLICATION_NAME AS APP_NAME,
       WORKLOAD_OCCURRENCE_STATE AS WL_STATE
FROM TABLE(MON_GET_CONNECTION(NULL,-2))
WHERE APPLICATION_HANDLE = 65564
```

This query returns the following output:

| SYSTEM_AUTH_ID | APP_NAME | WL_STATE |
|----------------|----------|----------|
| ZURBIE | db2bp | UOWWAIT |

MON_GET_APPLICATION_HANDLE - Get connection application handle

The MON_GET_APPLICATION_HANDLE scalar function returns the application handle of the invoking application.

Syntax

►►—MON_GET_APPLICATION_HANDLE—(—)—————►►

The schema is SYSPROC.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Information returned

The data type of the result is BIGINT.

Example

The MON_GET_APPLICATION_HANDLE scalar function can be used to pass in the application handle of the current session into monitoring functions which filter based on the application handle, such that the session can access its own monitoring information. For example:

```
select application_handle, application_name, application_id, member, rows_read
from table(sysproc.mon_get_connection(sysproc.mon_get_application_handle(), -1))
as conn
```

The following is an example of output from this query.

| APPLICATION_HANDLE | APPLICATION_NAME | APPLICATION_ID | MEMBER |
|--------------------|------------------|------------------------------|--------|
| 644 | db2bp | *LOCAL.amurchis.110831180720 | 0 |

| ROWS_READ |
|-----------|
| 0 |

1 record(s) selected.

MON_GET_APPLICATION_ID - Get connection application ID

The MON_GET_APPLICATION_ID scalar function returns the application ID of the invoking application.

The value returned by the function is unique within a 100-year interval and valid only for the duration of the connection established before calling the function.

►►—MON_GET_APPLICATION_ID—(—)—————►►

The schema is SYSPROC.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Information returned

The data type of the result is VARCHAR(128).

Example

```
values sysproc.mon_get_application_id()
```

The following is an example of output from this query.

```
1
-----
*LOCAL.amurchis.110831180720
```

MON_GET_AUTO_MAINT_QUEUE table function - Get information about the automatic maintenance jobs

The MON_GET_AUTO_MAINT_QUEUE table function returns information about all automatic maintenance jobs (with the exception of real-time statistics which does not submit jobs on the automatic maintenance queue) that are currently queued for execution by the autonomic computing daemon (db2acd).

Syntax

►►—MON_GET_AUTO_MAINT_QUEUE—(—)—————►►

The schema is SYSPROC.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Examples

Display the current jobs on the automatic maintenance queue on each partition.

```
SELECT MEMBER
      QUEUE_POSITION,
      JOB_STATUS,
      JOB_TYPE,
      VARCHAR(DB_NAME, 10) AS DB_NAME,
      OBJECT_TYPE,
      VARCHAR(OBJECT_SCHEMA, 10) AS OBJECT_SCHEMA
      VARCHAR(OBJECT_NAME, 10) AS OBJECT_NAME
FROM TABLE(MON_GET_AUTO_MAINT_QUEUE()) AS T
ORDER BY MEMBER, QUEUE_POSITION ASC
```

The following is an example of output from this query.

| MEMBER | QUEUE_POSITION | JOB_STATUS | JOB_TYPE | DB_NAME | OBJECT_TYPE | OBJECT_SCHEMA |
|--------|----------------|------------|----------|---------|-------------|---------------|
| 0 | 1 | EXECUTING | RUNSTATS | SAMPLE | TABLE | TEST |
| 0 | 2 | QUEUED | REORG | SAMPLE | TABLE | TEST |
| 0 | 3 | QUEUED | REORG | SAMPLE | TABLE | TEST |

| OBJECT_NAME |
|-------------|
| EMPLOYEE |
| T1 |
| BLAH |

3 record(s) selected.

Display the current jobs on the automatic maintenance queue.

```
SELECT JOB_STATUS,
      JOB_TYPE,
      OBJECT_TYPE,
      VARCHAR(OBJECT_NAME, 10) AS OBJECT_NAME,
      VARCHAR(JOB_DETAILS,60) AS JOB_DETAILS
FROM TABLE(MON_GET_AUTO_MAINT_QUEUE()) AS T
ORDER BY MEMBER, QUEUE_POSITION ASC
```

The following is an example of output from this query.

| JOB_STATUS | JOB_TYPE | OBJECT_TYPE | OBJECT_NAME |
|------------|----------|-------------|-------------|
| EXECUTING | REORG | TABLE | TP3 |

| JOB_DETAILS |
|--|
| REORG INDEXES ALLOW WRITE CLEANUP ALL; RECLAIM EXTENTS |

1 record(s) selected.

Usage notes

The MON_GET_AUTO_MAINT_QUEUE table function returns information about all queued and executing automatic maintenance jobs on all members. Note that there is a separate automatic maintenance queue per member.

Note: Jobs in the automatic maintenance queue are ordered first by earliest start time (such as start of next maintenance window where job may run), priority (for entries with same earliest start time) and queue entry time (for entries with the same earliest start time and priority).

The information returned from MON_GET_AUTO_MAINT_QUEUE supplements the information from the HEALTH_DB_HIC interface. HEALTH_DB_HIC shows the automatic maintenance status for each table to which automatic maintenance is applied, as well as the last time the status was updated (such as last time table was checked to see if maintenance is needed). The MON_GET_AUTO_MAINT_QUEUE interface provides a drill down for when the state is AUTOMATED, providing details about where the maintenance job is in the auto maintenance queue, and what other jobs are ahead of the job in the queue.

The MON_GET_AUTO_MAINT_QUEUE table function does not report any automatic maintenance jobs if automatic maintenance is not enabled.

Information returned

Table 113. Information returned for MON_GET_AUTO_MAINT_QUEUE

| Column Name | Data Type | Description |
|---------------|--------------|--|
| DB_NAME | VARCHAR(128) | db_name - Database name monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| OBJECT_TYPE | VARCHAR(8) | Type of object. One of: <ul style="list-style-type: none"> • DATABASE • NICKNAME • TABLE • VIEW |
| OBJECT_SCHEMA | VARCHAR(128) | object_schema - Object schema monitor element |
| OBJECT_NAME | VARCHAR(128) | object_name - Object name monitor element |
| JOB_TYPE | VARCHAR(12) | Type of automatic maintenance job. One of:
RUNSTATS
REORG
BACKUP |
| JOB_DETAILS | VARCHAR(256) | Details about the maintenance job if type is RUNSTATS or REORG. For RUNSTATS, indicates if the runstats job is doing a full runstats or just sampling. For REORG, lists the keywords that will be applied to modify the behavior of the REORG utility (for example, INDEXES, CLEANUP, and so on). For an index REORG, if the keywords indicate both CLEANUP and RECLAIM EXTENTS separated by a semicolon, index reorg cleanup is done followed by the evaluation of and potential run of index reclaim extents (reclaim extents is done if the evaluation of reclaimable space compared against the reclaimExtentsSizeForIndexObjects setting indicates reclaim is necessary). |
| JOB_STATUS | VARCHAR(10) | Current status of the job. One of:
QUEUED
EXECUTING |

Table 113. Information returned for MON_GET_AUTO_MAINT_QUEUE (continued)

| Column Name | Data Type | Description |
|----------------------|------------|--|
| JOB_PRIORITY | INTEGER | Numeric priority of job in the queue. Priority is only important for jobs with the same value for EARLIEST_START_TIME. |
| MAINT_WINDOW_TYPE | VARCHAR(8) | Indicates which type of maintenance window will be used for the job. One of:

ONLINE

OFFLINE |
| QUEUE_POSITION | INTEGER | Indicates the position of the job in the automatic maintenance queue. |
| QUEUE_ENTRY_TIME | TIMESTAMP | Time that the job was added to the automatic maintenance queue. |
| EXECUTION_START_TIME | TIMESTAMP | Time that job started execution, if status is EXECUTING. NULL otherwise. |
| EARLIEST_START_TIME | TIMESTAMP | Start time of next maintenance window where job is eligible to run. |

MON_GET_AUTO_RUNSTATS_QUEUE table function - Retrieve information about objects queued for evaluation

The MON_GET_AUTO_RUNSTATS_QUEUE table function returns information about all objects which are currently queued for evaluation by automatic statistics collection in the currently connected database.

Syntax

►►—MON_GET_AUTO_RUNSTATS_QUEUE—()—◄◄

The schema is SYSPROC.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

Display all objects that are currently under evaluation by automatic statistics collection in the currently connected database.

```
SELECT QUEUE_POSITION,
       OBJECT_TYPE,
       OBJECT_STATUS,
```

```

        VARCHAR(OBJECT_SCHEMA, 10) AS OBJECT_SCHEMA,
        VARCHAR(OBJECT_NAME, 10) AS OBJECT_NAME
FROM TABLE(MON_GET_AUTO_RUNSTATS_QUEUE()) AS T
ORDER BY QUEUE_POSITION ASC

```

The following is an example of output from this query.

| QUEUE_POSITION | OBJECT_TYPE | OBJECT_STATUS | OBJECT_SCHEMA | OBJECT_NAME |
|----------------|-------------|--------------------|---------------|-------------|
| 1 | TABLE | JOB_SUBMITTED | TEST | EMPLOYEE |
| 2 | TABLE | EVALUATION_PENDING | TEST | T1 |
| 3 | TABLE | EVALUATION_PENDING | TEST | BLAH |

3 record(s) selected.

Usage notes

The MON_GET_AUTO_RUNSTATS_QUEUE table function returns information about all objects queued for evaluation by automatic statistics collection in the currently connected database. Objects queued for evaluation will be examined by automatic statistics collection to determine if a statistics update is required. When an object is found to require statistics update, a job is submitted to the automatic maintenance scheduler and evaluation temporarily blocks until the job completes execution. Use the MON_GET_AUTO_MAINT_QUEUE interface to monitor the progress of the statistics collection job. After an object has been evaluated by automatic statistics collection is removed from the evaluation queue, it will not be added to the queue again until the next automatic statistics collection evaluation interval. For a per-object history of evaluation, refer to the HEALTH_DB_HIC interface.

Each database has a separate automatic statistics collection evaluation queue. The MON_GET_AUTO_RUNSTATS_QUEUE reports the evaluation queue for the currently connected database.

The list reported by the MON_GET_AUTO_RUNSTATS_QUEUE interface may be empty if there are no objects currently queued for evaluation by automatic statistics collection. For example, if automatic statistics collection has processed all the tables that were queued for evaluation in the current evaluation interval, the list of objects queued for evaluation will be empty until the next evaluation interval. Automatic statistics collection evaluation intervals occur roughly every two hours.

The list reported by the MON_GET_AUTO_RUNSTATS_QUEUE interface is also empty if automatic statistics collection is not enabled.

Information returned

Table 114. Information returned for MON_GET_AUTO_RUNSTATS_QUEUE

| Column Name | Data Type | Description |
|---------------|--------------|---|
| OBJECT_TYPE | VARCHAR(8) | Type of object. One of: <ul style="list-style-type: none"> NICKNAME TABLE VIEW |
| OBJECT_SCHEMA | VARCHAR(128) | object_schema - Object schema monitor element |
| OBJECT_NAME | VARCHAR(128) | object_name - Object name monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

Table 114. Information returned for MON_GET_AUTO_RUNSTATS_QUEUE (continued)

| Column Name | Data Type | Description |
|------------------|-------------|--|
| OBJECT_STATUS | VARCHAR(18) | Status of queued object. One of:

EVALUATION_PENDING - Automatic statistics collection needs to look at the object to determine if a statistics update is required.

JOB_SUBMITTED - Automatic statistics collection has determined that a statistics update is required for an object and has submitted a job to the automatic maintenance scheduler. Automatic statistics collection is waiting for the job to complete. |
| JOB_SUBMIT_TIME | TIMESTAMP | If the status is JOB_SUBMITTED, the time at which the automatic statistics collection job was submitted to the automatic maintenance scheduler. NULL otherwise. |
| QUEUE_POSITION | INTEGER | Indicates the position of the job in the automatic statistics collection evaluation queue. |
| QUEUE_ENTRY_TIME | TIMESTAMP | Time that the job was added to the automatic statistics collection evaluation queue |

MON_GET_BUFFERPOOL table function - Get buffer pool metrics

The MON_GET_BUFFERPOOL table function returns monitor metrics for one or more buffer pools.

Syntax

►► MON_GET_BUFFERPOOL (—bp_name—, —member—) ◀◀

The schema is SYSPROC.

Table function parameters

bp_name

An input argument of type VARCHAR(128) that specifies a valid buffer pool name in the currently connected database when calling this function. If the argument is null or an empty string, metrics are retrieved for all buffer pools in the database.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

Compute the buffer pool hit ratio.

```
WITH BPMETRICS AS (  
  SELECT bp_name,  
         pool_data_l_reads + pool_temp_data_l_reads +  
         pool_index_l_reads + pool_temp_index_l_reads +  
         pool_xda_l_reads + pool_temp_xda_l_reads as logical_reads,  
         pool_data_p_reads + pool_temp_data_p_reads +  
         pool_index_p_reads + pool_temp_index_p_reads +  
         pool_xda_p_reads + pool_temp_xda_p_reads as physical_reads,  
         member  
  FROM TABLE(MON_GET_BUFFERPOOL('',-2)) AS METRICS)  
SELECT  
  VARCHAR(bp_name,20) AS bp_name,  
  logical_reads,  
  physical_reads,  
  CASE WHEN logical_reads > 0  
    THEN DEC((1 - (FLOAT(physical_reads) / FLOAT(logical_reads))) * 100,5,2)  
    ELSE NULL  
  END AS HIT_RATIO,  
  member  
FROM BPMETRICS;
```

The following is an example of output from this query.

| BP_NAME | LOGICAL_READS | PHYSICAL_READS | HIT_RATIO | MEMBER |
|----------------|---------------|----------------|-----------|--------|
| IBMDEFAULTBP | 619 | 385 | 37.80 | 0 |
| IBMSYSTEMBP4K | 0 | 0 | - | 0 |
| IBMSYSTEMBP8K | 0 | 0 | - | 0 |
| IBMSYSTEMBP16K | 0 | 0 | - | 0 |
| IBMSYSTEMBP32K | 0 | 0 | - | 0 |

5 record(s) selected.

Usage notes

The MON_GET_BUFFERPOOL table function returns one row of data per database buffer pool and per database member. No aggregation across database members is performed. However, aggregation can be achieved through SQL queries as shown in the example.

Metrics collected by this function are controlled at the database level using the mon_obj_metrics configuration parameter. By default, metrics collection is enabled.

Information returned

Table 115. Information returned for MON_GET_BUFFERPOOL

| Column Name | Data Type | Description |
|------------------|--------------|---|
| BP_NAME | VARCHAR(128) | |
| MEMBER | SMALLINT | member - Database member |
| AUTOMATIC | SMALLINT | automatic - Buffer pool automatic |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |

Table 115. Information returned for MON_GET_BUFFERPOOL (continued)

| Column Name | Data Type | Description |
|----------------------------|-----------|---|
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer pool XDA data logical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer pool XDA data physical reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer pool XDA data writes |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer pool asynchronous XDA data reads |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer pool asynchronous XDA read requests |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer pool asynchronous XDA data writes |
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer pool no victim buffers |
| POOL_LSN_GAP_CLNS | BIGINT | pool_lsn_gap_clns - Buffer pool log space cleaners triggered |
| POOL_DRTY_PG_STEAL_CLNS | BIGINT | pool_drty_pg_steal_clns - Buffer pool victim page cleaners triggered |

Table 115. Information returned for MON_GET_BUFFERPOOL (continued)

| Column Name | Data Type | Description |
|------------------------------------|-----------|--|
| POOL_DRTY_PG_THRSH_CLNS | BIGINT | pool_drtty_pg_thrsh_clns - Buffer pool threshold cleaners triggered |
| VECTORED_IOS | BIGINT | vectored_ios - Number of vectored IO requests |
| PAGES_FROM_VECTORED_IOS | BIGINT | pages_from_vectored_ios - Total number of pages read by vectored IO |
| BLOCK_IOS | BIGINT | block_ios - Number of block IO requests |
| PAGES_FROM_BLOCK_IOS | BIGINT | pages_from_block_ios - Total number of pages read by block IO |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| FILES_CLOSED | BIGINT | files_closed - Database files closed |
| POOL_DATA_GBP_L_READS | BIGINT | pool_data_gbp_l_reads - Group buffer pool data logical reads monitor element |
| POOL_DATA_GBP_P_READS | BIGINT | pool_data_gbp_p_reads - Group buffer pool data physical reads monitor element |
| POOL_DATA_LBP_PAGES_FOUND | BIGINT | pool_data_lbp_pages_found - Local buffer pool found data pages monitor element |
| POOL_DATA_GBP_INVALID_PAGES | BIGINT | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages monitor element |
| POOL_INDEX_GBP_L_READS | BIGINT | pool_index_gbp_l_reads - Group buffer pool index logical reads monitor element |
| POOL_INDEX_GBP_P_READS | BIGINT | pool_index_gbp_p_reads - Group buffer pool index physical reads monitor elements |
| POOL_INDEX_LBP_PAGES_FOUND | BIGINT | pool_index_lbp_pages_found - Local buffer pool index pages found monitor element |
| POOL_INDEX_GBP_INVALID_PAGES | BIGINT | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages |
| POOL_ASYNC_DATA_GBP_L_READS | BIGINT | pool_async_data_gbp_l_reads - Asynchronous group buffer pool data logical reads |
| POOL_ASYNC_DATA_GBP_P_READS | BIGINT | pool_async_data_gbp_p_reads - Asynchronous group buffer pool data physical reads |
| POOL_ASYNC_DATA_LBP_PAGES_FOUND | BIGINT | pool_async_data_lbp_pages_found - Asynchronous local buffer pool data pages found |
| POOL_ASYNC_DATA_GBP_INVALID_PAGES | BIGINT | pool_async_data_gbp_invalid_pages - Asynchronous group buffer pool invalid data pages |
| POOL_ASYNC_INDEX_GBP_L_READS | BIGINT | pool_async_index_gbp_l_reads - Asynchronous group buffer pool index logical reads |
| POOL_ASYNC_INDEX_GBP_P_READS | BIGINT | pool_async_index_gbp_p_reads - Asynchronous group buffer pool index physical reads |
| POOL_ASYNC_INDEX_LBP_PAGES_FOUND | BIGINT | pool_async_index_lbp_pages_found - Asynchronous local buffer pool index pages found |
| POOL_ASYNC_INDEX_GBP_INVALID_PAGES | BIGINT | pool_async_index_gbp_invalid_pages - Asynchronous group buffer pool invalid index pages |
| POOL_XDA_GBP_L_READS | BIGINT | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |
| POOL_XDA_GBP_P_READS | BIGINT | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |
| POOL_XDA_LBP_PAGES_FOUND | BIGINT | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |
| POOL_XDA_GBP_INVALID_PAGES | BIGINT | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| POOL_ASYNC_XDA_GBP_L_READS | BIGINT | pool_async_xda_gbp_l_reads - Group buffer pool XDA data asynchronous logical read requests |

Table 115. Information returned for MON_GET_BUFFERPOOL (continued)

| Column Name | Data Type | Description |
|------------------------------------|-----------|--|
| POOL_ASYNC_XDA_GBP_P_READS | BIGINT | pool_async_xda_gbp_p_reads - Group buffer pool XDA data asynchronous physical read requests |
| POOL_ASYNC_XDA_LBP_PAGES_FOUND | BIGINT | pool_async_xda_lbp_pages_found - Asynchronous local buffer pool XDA data pages found |
| POOL_ASYNC_XDA_GBP_INVALID_PAGES | BIGINT | pool_async_xda_gbp_invalid_pages - Asynchronous group buffer pool invalid XDA data pages |
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer Pool Asynchronous Read Time |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time |
| BP_CUR_BUFFSZ | BIGINT | bp_cur_buffsz - Current Size of Buffer Pool |
| POOL_QUEUED_ASYNC_DATA_REQS | BIGINT | pool_queued_async_data_reqs - Data prefetch requests |
| POOL_QUEUED_ASYNC_INDEX_REQS | BIGINT | pool_queued_async_index_reqs - Index prefetch requests |
| POOL_QUEUED_ASYNC_XDA_REQS | BIGINT | pool_queued_async_xda_reqs - XDA prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_OTHER_REQS | BIGINT | pool_queued_async_other_reqs - Non-prefetch requests |
| POOL_QUEUED_ASYNC_DATA_PAGES | BIGINT | pool_queued_async_data_pages - Data pages prefetch requests |
| POOL_QUEUED_ASYNC_INDEX_PAGES | BIGINT | pool_queued_async_index_pages - Index pages prefetch requests |
| POOL_QUEUED_ASYNC_XDA_PAGES | BIGINT | pool_queued_async_xda_pages - XDA pages prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES | BIGINT | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES | BIGINT | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_DATA_REQS | BIGINT | pool_failed_async_data_reqs - Failed data prefetch requests |
| POOL_FAILED_ASYNC_INDEX_REQS | BIGINT | pool_failed_async_index_reqs - Failed index prefetch requests |
| POOL_FAILED_ASYNC_XDA_REQS | BIGINT | pool_failed_async_xda_reqs - Failed XDA prefetch requests |
| POOL_FAILED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_OTHER_REQS | BIGINT | pool_failed_async_other_reqs - Failed non-prefetch requests |
| SKIPPED_PREFETCH_DATA_P_READS | BIGINT | skipped_prefetch_data_p_reads - Skipped prefetch data physical reads |
| SKIPPED_PREFETCH_INDEX_P_READS | BIGINT | skipped_prefetch_index_p_reads - Skipped prefetch index physical reads |

Table 115. Information returned for MON_GET_BUFFERPOOL (continued)

| Column Name | Data Type | Description |
|---|-----------|---|
| SKIPPED_PREFETCH_XDA_P_READS | BIGINT | skipped_prefetch_xda_p_reads - Skipped prefetch XDA physical reads |
| SKIPPED_PREFETCH_TEMP_DATA_P_READS | BIGINT | skipped_prefetch_temp_data_p_reads - Skipped prefetch temporary data physical reads |
| SKIPPED_PREFETCH_TEMP_INDEX_P_READS | BIGINT | skipped_prefetch_temp_index_p_reads - Skipped prefetch temporary index physical reads |
| SKIPPED_PREFETCH_TEMP_XDA_P_READS | BIGINT | skipped_prefetch_temp_xda_p_reads - Skipped prefetch temporary XDA data physical reads |
| SKIPPED_PREFETCH_UOW_DATA_P_READS | BIGINT | skipped_prefetch_uow_data_p_reads - Skipped prefetch unit of work data physical reads |
| SKIPPED_PREFETCH_UOW_INDEX_P_READS | BIGINT | skipped_prefetch_uow_index_p_reads - Skipped prefetch unit of work index physical reads |
| SKIPPED_PREFETCH_UOW_XDA_P_READS | BIGINT | skipped_prefetch_uow_xda_p_reads - Skipped prefetch unit of work XDA data physical reads |
| SKIPPED_PREFETCH_UOW_TEMP_DATA_P_READS | BIGINT | skipped_prefetch_uow_temp_data_p_reads - Skipped prefetch unit of work temporary data physical reads |
| SKIPPED_PREFETCH_UOW_TEMP_INDEX_P_READS | BIGINT | skipped_prefetch_uow_temp_index_p_reads - Skipped prefetch unit of work temporary index physical reads |
| SKIPPED_PREFETCH_UOW_TEMP_XDA_P_READS | BIGINT | skipped_prefetch_uow_temp_xda_p_reads - Skipped prefetch unit of work temporary XDA data physical reads |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| PREFETCH_WAITS | BIGINT | prefetch_waits - Prefetcher wait count |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |
| POOL_ASYNC_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_async_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found by asynchronous EDUs in a local buffer pool monitor element monitor element |
| POOL_ASYNC_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_async_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found by asynchronous EDUs in a local buffer pool monitor element monitor element |
| POOL_ASYNC_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_async_xda_gbp_indep_pages_found_in_lbp - Group buffer pool independent XML storage object(XDA) pages found by asynchronous EDUs in a local buffer pool monitor element monitor element |
| POOL_COL_L_READS | BIGINT | pool_col_l_reads - Buffer pool column-organized logical reads monitor element |
| POOL_TEMP_COL_L_READS | BIGINT | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| POOL_COL_P_READS | BIGINT | pool_col_p_reads - Buffer pool column-organized physical reads |
| POOL_TEMP_COL_P_READS | BIGINT | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| POOL_COL_LBP_PAGES_FOUND | BIGINT | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found monitor element |
| POOL_COL_WRITES | BIGINT | pool_col_writes - Buffer pool column-organized writes |

Table 115. Information returned for MON_GET_BUFFERPOOL (continued)

| Column Name | Data Type | Description |
|---|-----------|---|
| POOL_ASYNC_COL_READS | BIGINT | pool_async_col_reads - Buffer pool asynchronous column-organized reads |
| POOL_ASYNC_COL_READ_REQS | BIGINT | pool_async_col_read_reqs - Buffer pool asynchronous column-organized read requests |
| POOL_ASYNC_COL_WRITES | BIGINT | pool_async_col_writes - Buffer pool asynchronous column-organized writes |
| POOL_ASYNC_COL_LBP_PAGES_FOUND | BIGINT | pool_async_col_lbp_pages_found - Asynchronous local buffer pool column-organized pages found |
| POOL_COL_GBP_L_READS | BIGINT | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| POOL_COL_GBP_P_READS | BIGINT | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| POOL_COL_GBP_INVALID_PAGES | BIGINT | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |
| POOL_ASYNC_COL_GBP_L_READS | BIGINT | pool_async_col_gbp_l_reads - Asynchronous group buffer pool column-organized logical reads |
| POOL_ASYNC_COL_GBP_P_READS | BIGINT | pool_async_col_gbp_p_reads - Asynchronous group buffer pool column-organized physical reads |
| POOL_ASYNC_COL_GBP_INVALID_PAGES | BIGINT | pool_async_col_gbp_invalid_pages - Asynchronous group buffer pool invalid data pages monitor element |
| POOL_ASYNC_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_async_col_gbp_indep_pages_found_in_lbp - Asynchronous group buffer pool column-organized independent data pages found in local buffer pool monitor element |
| POOL_QUEUED_ASYNC_COL_REQS | BIGINT | pool_queued_async_col_reqs - Column-organized prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_REQS | BIGINT | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| POOL_QUEUED_ASYNC_COL_PAGES | BIGINT | pool_queued_async_col_pages - Column-organized page prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_PAGES | BIGINT | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| POOL_FAILED_ASYNC_COL_REQS | BIGINT | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| POOL_FAILED_ASYNC_TEMP_COL_REQS | BIGINT | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |
| SKIPPED_PREFETCH_COL_P_READS | BIGINT | skipped_prefetch_col_p_reads - Skipped prefetch column-organized physical reads |
| SKIPPED_PREFETCH_TEMP_COL_P_READS | BIGINT | skipped_prefetch_temp_col_p_reads - Skipped prefetch column-organized temporary physical reads |
| SKIPPED_PREFETCH_UOW_COL_P_READS | BIGINT | skipped_prefetch_uow_col_p_reads - Skipped prefetch unit of work column-organized physical reads |
| SKIPPED_PREFETCH_UOW_TEMP_COL_P_READS | BIGINT | skipped_prefetch_uow_temp_col_p_reads - Skipped prefetch unit of work column-organized temporary physical reads |
| BP_PAGES_LEFT_TO_REMOVE | BIGINT | bp_pages_left_to_remove - Number of Pages Left to Remove monitor element |
| BP_TBSP_USE_COUNT | BIGINT | bp_tbsp_use_count - Number of Table Spaces Mapped to Buffer Pool monitor element |
| POOL_DATA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |

Table 115. Information returned for MON_GET_BUFFERPOOL (continued)

| Column Name | Data Type | Description |
|---|-----------|--------------------------|
| POOL_INDEX_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_READ_TIME | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_WRITE_TIME | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_READS | BIGINT | Reserved for future use. |

Table 115. Information returned for MON_GET_BUFFERPOOL (continued)

| Column Name | Data Type | Description |
|---|-----------|--|
| POOL_ASYNC_INDEX_CACHING_TIER_READS | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_READS | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_READS | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| LOB_PREFETCH_WAIT_TIME | BIGINT | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| LOB_PREFETCH_REQS | BIGINT | lob_prefetch_reqs - LOB prefetcher request count monitor element |

MON_GET_CF table function - Get cluster caching facility metrics

The MON_GET_CF table function returns status information about one or more cluster caching facilities (also known as CFs) on the system.

Syntax

►►—MON_GET_CF—(—*id*—)—————►►

The schema is SYSPROC.

Table function parameters

id An input argument of type INTEGER that specifies the identifier of the cluster caching facility for which data will be returned. You can use the DB2_CF administrative view to obtain the identifiers of the cluster caching facilities on your system. If this parameter is NULL, information for all cluster caching facilities on the system is returned.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Examples

The following query displays information about the group buffer pool size for all cluster caching facilities on the system. It shows a resize in progress for the group buffer pool. The CURRENT_CF_GBP_SIZE parameter shows the group buffer pool memory currently in use. The CONFIGURED_CF_GBP_SIZE parameter shows the group buffer pool memory currently allocated and reserved. The TARGET_CF_GBP_SIZE parameter shows the group buffer pool dynamic resize target.

```
SELECT SUBSTR(HOST_NAME,1,8) AS HOST,
       SUBSTR(DB_NAME, 1,8) AS DBNAME,
       CURRENT_CF_GBP_SIZE,
       CONFIGURED_CF_GBP_SIZE,
       TARGET_CF_GBP_SIZE
FROM TABLE( MON_GET_CF( cast(NULL as integer) ) ) AS CAMETRICS
ORDER BY HOST;
```

The following is an example of output from this query.

| HOST | DBNAME | CURRENT_CF_GBP_SIZE | CONFIGURED_CF_GBP_SIZE | TARGET_CF_GBP_SIZE |
|------|--------|---------------------|------------------------|--------------------|
| cf15 | SAMPLE | 2402 | 3000 | 5000 |
| cf16 | SAMPLE | 1276 | 3000 | 5000 |

The next example returns information about the group buffer pool size for the cluster caching facility that has the identifier of 128.

```
SELECT SUBSTR(HOST_NAME,1,8) AS HOST,
       SUBSTR(DB_NAME, 1,8) AS DBNAME,
       CURRENT_CF_GBP_SIZE,
       CONFIGURED_CF_GBP_SIZE,
       TARGET_CF_GBP_SIZE
FROM TABLE( MON_GET_CF( 128 ) ) AS CAMETRICS;
```

The following is an example of output from this query.

```
HOST  DBNAME  CURRENT_CF_GBP_SIZE  CONFIGURED_CF_GBP_SIZE  TARGET_CF_GBP_SIZE
-----
cf16  SAMPLE              1276                  2000                  2000
```

Usage notes

The MON_GET_CF table function returns one row of data per cluster caching facility-database pairing defined on the instance. No aggregation across cluster caching facilities or databases is performed.

Information returned

Table 116. Information returned for MON_GET_CF

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|---|
| HOST_NAME | VARCHAR(128) | host_name - Host name |
| ID | SMALLINT | id - Identification |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| CURRENT_CF_GBP_SIZE | INTEGER | current_cf_gbp_size - CurrentCF group buffer pool size |
| CONFIGURED_CF_GBP_SIZE | INTEGER | configured_cf_gbp_size - Configured CF group buffer pool size |
| TARGET_CF_GBP_SIZE | INTEGER | target_cf_gbp_size - Target CF group buffer pool size |
| CURRENT_CF_LOCK_SIZE | INTEGER | current_cf_lock_size - CurrentCF lock size |
| CONFIGURED_CF_LOCK_SIZE | INTEGER | configured_cf_lock_size - Configured CF lock size |
| TARGET_CF_LOCK_SIZE | INTEGER | target_cf_lock_size - Target CF lock size |
| CURRENT_CF_SCA_SIZE | INTEGER | current_cf_sca_size - Current CF shared communication area size |
| CONFIGURED_CF_SCA_SIZE | INTEGER | configured_cf_sca_size - Configured CF shared communication area size |
| TARGET_CF_SCA_SIZE | INTEGER | target_cf_sca_size - Target CF shared communication area size |
| CURRENT_CF_MEM_SIZE | INTEGER | current_cf_mem_size - CurrentCF memory size |
| CONFIGURED_CF_MEM_SIZE | INTEGER | configured_cf_mem_size - Configured CF memory size |

MON_GET_CF_CMD - Get processing times for cluster caching facility commands

The MON_GET_CF_CMD table function returns information about the processing times for cluster caching facility (CF) commands.

Syntax

►►—MON_GET_CF_CMD—(—*id*—)—————►►

The schema is SYSPROC.

Table function parameters

id An input argument of type INTEGER that specifies the identifier of the CF for which data is returned. You can use the DB2_CF administrative view to obtain the identifiers for the CFs in your Db2 pureScale environment. If this parameter is NULL, information is returned for all CFs in your Db2 pureScale environment.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

The following statement is issued on a Db2 pureScale instance with multiple members and two cluster caching facilities with identifiers CF15 and CF16:

```
SELECT SUBSTR(HOSTNAME,1,8) AS HOST,  
       SUBSTR(CF_CMD_NAME,1,20) AS CF_CMD_NAME,  
       TOTAL_CF_REQUESTS AS REQUESTS  
FROM TABLE(MON_GET_CF_CMD(NULL) ) AS CFMETRICS  
WHERE CF_CMD_NAME = 'SetLockState' OR CF_CMD_NAME = 'ReadAndRegister'
```

An example of output from this query is as follows:

| HOST | CF_CMD_NAME | REQUESTS |
|-------|-----------------|----------|
| ----- | ----- | ----- |
| CF15 | SetLockState | 4 |
| CF15 | ReadAndRegister | 7 |
| CF16 | SetLockState | 1 |
| CF16 | ReadAndRegister | 1 |

Usage notes

Data is returned only if you issue this function in a Db2 pureScale environment.

The reported values are provided by the CF and are independent of the instance level for a member or the number of databases that are active for a member.

The values that are returned in the **TOTAL_CF_REQUESTS** and **TOTAL_CF_CMD_TIME_MICRO** columns are calculated since cluster startup. You cannot reset the values; they are read-only.

You can calculate the average time by dividing the return value of the **TOTAL_CF_CMD_TIME_MICRO** column by the number returned by the **TOTAL_CF_REQUESTS** column.

Information returned

Table 117. Information that is returned for the **MON_GET_CF_CMD** function

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|--|
| HOSTNAME | VARCHAR(255) | hostname - Host name monitor element |
| ID | SMALLINT | id - Identification |
| CF_CMD_NAME | VARCHAR(128) | Name of the CF command. |
| TOTAL_CF_REQUESTS | BIGINT | Total number of requests in the CF for this command. |
| TOTAL_CF_CMD_TIME_MICRO | BIGINT | Total processing time in the CF for this command, in microseconds. |

Table 118. List of CF commands that are monitored by the **MON_GET_CF_CMD** function

| CF Command or alias name | Description |
|--|--|
| AllocationUnitRecovery | Recover a global lock manager (GLM) after a local lock manager (LLM) detaches, releasing all non-retained locks and downgrading retained locks. |
| AttachLocalCache | Attach to the group buffer pool (GBP). |
| CrossInvalidate XI | Send a single cross-invalidate message. |
| ExtendedMessageResponseBlock | Return the metric for a large message response block transmission along with the acknowledgment for the response transmission from the terminating side, as measured over the communication medium. |
| ExtendedMessageResponseBlockAsync | Return the metric for a large message response block transmission over the communication medium only. There is no waiting for an acknowledgment for the response transmission from the terminating side over the communication medium; completion of the transmission is checked separately. |
| LockNotification | Send a lock notification. |
| MessageResponseBlock | Return the metric for a message response block transmission along with the acknowledgment for the response transmission from the terminating side, as measured over the communication medium. |
| MessageResponseBlockAsync | Return the metric for a message response block transmission over the communication medium only. There is no waiting for an acknowledgment for the response transmission from the terminating side over the communication medium; completion of the transmission is checked separately. |
| ProcessSetLockStateExistingClient | Process the set lock state from an existing client. |
| ProcessSetLockStateExistingLock | Process the set lock state time for an existing lock. |
| ProcessSetLockStateNewClient | Process the set lock state from a new client. |

Table 118. List of CF commands that are monitored by the MON_GET_CF_CMD function (continued)

| CF Command or alias name | Description |
|---|--|
| ProcessSetLockStateNewLock | Process the set lock state time for a new lock. |
| ProcessSetLockStateSingleLock | Process the set lock state internal time for a single lock. |
| ReadAndRegister RAR | Read a single page from the CF. |
| ReadCacheInfo | Read the GBP cache information. |
| ReadCastoutClass | Read the list of pages in the castout class that are eligible to be cast out from the CF. |
| ReadCCInfo | Read the castout information for a range of castout classes from the CF. |
| ReadForCastout | Read data from the CF so that it can be cast out. |
| ReadForCastoutMultiple | Scan the castout queue in the CF for eligible entries. This is a combination of the ReadCastoutClass and ReadForCastout commands. |
| ReadForCastoutMultipleList | Scan the castout queue in the CF for eligible entries. This is a combination of the ReadCastoutClass and ReadForCastout commands for a specified list. |
| ReadLocks | Read the locks that are defined in the GLM structure in the CF. |
| ReadSA | Read the aggregate value from a smart array (SA) in the CF. |
| ReadSetLFS | Get or set a log flush sequence (LFS) number in the CF. |
| RecordLockState RLS | Record the lock state to the secondary CF. |
| RegisterPageList | Register a list of pages with the CF. |
| ReleaseCastoutLocks | Release castout locks on a page in the CF. |
| SetLockState SLS | Set the lock state of the CF. |
| TestPageValidity | Test whether the local page is valid in the CF. |
| TryInstant | Return the commands that are issued to obtain an instant lock from the CF. |
| WriteAndRegister WAR | Send a single-page image into the CF. |
| WriteAndRegisterMultiple WARM | Send a multiple-pages image into the CF. |
| WriteAndRegisterMultipleSubOperation | Suboperation of sending a multiple-pages image into the CF. This command returns the processing time of each page in the CF after the data was read from the communications medium for a WARM operation. |
| WriteSA | Write a value into an SA and retrieve the aggregate value in the CF. |

MON_GET_CF_WAIT_TIME - Get wait times for cluster caching facility commands

The MON_GET_CF_WAIT_TIME table function returns the total time measurement for cluster caching facility (CF) commands.

The time measurement returned by this function includes the network transport time to and from the CF and the execution time of the command within the CF.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_GET_CF_WAIT_TIME—(—*member*—)————►►

Table function parameters

The schema is SYSPROC.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database. Specify -1 for the current database member or -2 for all active members. If you specify the NULL value, -1 is set implicitly.

Information returned

Table 119. Information that is returned for the MON_GET_CF_WAIT_TIME function

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|--------------|--|
| MEMBER | SMALLINT | member - Database member |
| HOSTNAME | VARCHAR(255) | hostname - Host name monitor element |
| ID | SMALLINT | id - Identification |
| CF_CMD_NAME | VARCHAR(128) | Name of the CF command. |
| TOTAL_CF_REQUESTS | BIGINT | Total number of requests in the CF for this command. |
| TOTAL_CF_WAIT_TIME_MICRO | BIGINT | Total wait time in the CF for this command, in microseconds. |

Table 120. List of CF commands that are monitored by the MON_GET_CF_WAIT_TIME function

| CF Command or alias name | Description |
|--------------------------|--|
| DeletePage | Delete a page from the CF. |
| DeletePageList | Delete a list of pages from the CF. |
| DeleteSA | Delete an entry from a smart array (SA) in the CF. |
| DeleteSAList | Delete a list of entries from an SA in the CF. |
| DeregisterPage | Deregister a page from the CF. |

Table 120. List of CF commands that are monitored by the MON_GET_CF_WAIT_TIME function (continued)

| CF Command or alias name | Description |
|--|---|
| DeregisterPageList | Deregister a list of pages from the CF. |
| GetAndIncLFS | Get and increment the log flush sequence (LFS) number in the CF. |
| GetLFS | Get an LFS from the CF. |
| GetLSN | Get the log sequence number (LSN) from the CF. |
| GetNotification | Return the commands issued to get notification messages from the global lock manager (GLM) in the CF. |
| GetStatus | Get the user control structures field from the CF. |
| GetUDF | Get the oldest page from the CF by using the user data field. |
| GLMDump | Request a dump of the GLM structure from the CF. |
| InitNotify | Return the commands issued to initialize message notification from the GLM structure in the CF. |
| MemberUD | Access the per member user data field in the CF. |
| ReadAndRegister RAR | Read a single page from the CF. |
| ReadAndRegisterMultiple RARM | Read multiple pages from the CF. |
| ReadCastoutClass | Read list of pages in the castout class that are eligible to be cast out from the CF. |
| ReadCCInfo | Read castout information for a range of castout classes from the CF. |
| ReadForCastout | Read data from the CF so that it can be cast out. |
| ReadForCastoutMultiple | Scan the castout queue in the CF for eligible entries. This command is a combination of the ReadCastoutClass and ReadForCastout commands. |
| ReadForCastoutMultipleList | Scan the castout queue in the CF for eligible entries. This command is a combination of the ReadCastoutClass and ReadForCastout) commands for a specified list. |
| ReadLocks | Read the locks that are defined in the GLM structure in the CF. |
| ReadSA | Read the aggregate value from an SA in the CF. |
| ReadSAList | Read all values for all SA variables or a particular SA variable in the CF. |
| ReleaseCastoutLocks | Release the castout lock on a page in the CF. |
| RecordLockState RLS | Record the lock state to the secondary CF. |
| RecordLockStateAsync | Record the lock state asynchronously to the secondary CF. |
| RecordLockStateMultiple RLSN | Record multiple lock states to the secondary CF. |
| RegisterPageList | Register a list of pages with the CF. |
| SetLFS | Set the LFS number in the CF. |
| SetLSN | Set the LSN in the CF. |
| SetLockState SLS | Set the lock state to the CF. |
| SetLockStateAsync | Set the lock state asynchronously to the CF. |

Table 120. List of CF commands that are monitored by the `MON_GET_CF_WAIT_TIME` function (continued)

| CF Command or alias name | Description |
|---|--|
| SetLockStateMultiple SLSN | Set multiple lock states to the CF. |
| SetLockStateMultipleAsync | Set multiple lock states asynchronously to the CF. |
| SetStatus | Set the user control structures field in the CF. |
| TestPageValidity | Test whether a local page is valid in the CF. |
| TryInstant | Return the commands issued to obtain an instant lock from the CF. |
| WriteAndRegister WAR | Send a single-page image into the CF. |
| WriteAndRegisterMultiple WARM | Send a multiple-pages image into the CF. |
| WriteSA | Write a value into an SA and retrieve the aggregate value in the CF. |
| WriteSAList | Write one or more SA variables in the CF. |

Usage notes

Data is returned only if you issue this function in a Db2 pureScale environment.

This table function returns one row of data per command and per member.

The values are provided at the instance level for a member, and are independent of the number of databases active for a member, because a member is connected to a CF at the instance level.

The values that are returned in the **TOTAL_CF_REQUESTS** and **TOTAL_CF_WAIT_TIME_MICRO** return columns are calculated since cluster startup. You cannot reset the values; they are read-only.

You can calculate the average time by dividing the return value of the **TOTAL_CF_WAIT_TIME_MICRO** column by the number returned by the **TOTAL_CF_REQUESTS** column.

Example

To examine the rate at which page deregistration occurs on a specific CF, which is identified in the `db2nodes.cfg` file as number 128, you want to compare values over a specific length of time. To begin, issue the following statement:

```
SELECT TOTAL_CF_WAIT_TIME_MICRO AS Dereg_T1,
       TOTAL_CF_REQUESTS AS Dereg_RQ1
FROM TABLE (MON_GET_CF_WAIT_TIME( -1 ))
WHERE CF_CMD_NAME = 'DeregisterPage' and ID = '128'
```

This query returns the initial values for the comparison. Then, wait 10 seconds for the CF to deregister additional pages.

To get the updated values, issue the following statement:

```
SELECT TOTAL_CF_WAIT_TIME_MICRO AS Dereg_T2,
       TOTAL_CF_REQUESTS AS Dereg_RQ2
FROM TABLE (MON_GET_CF_WAIT_TIME( -1 ))
WHERE CF_CMD_NAME = 'DeregisterPage' and ID = '128'
```

Finally, calculate the rate at which the CF is deregistrating pages by using the following formula:

$$\text{DEREG_PAGE_RATE} = \frac{(\text{DEREG_RQ2} - \text{DEREG_RQ1})}{(\text{DEREG_T2} - \text{DEREG_T1})}$$

MON_GET_CONNECTION table function - Get connection metrics

The MON_GET_CONNECTION table function returns metrics for one or more connections.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

```
►► MON_GET_CONNECTION (—application_handle—, —————►
►member [ , —system_appls— ] ) —————►◄
```

The schema is SYSPROC.

Routine parameters

application_handle

An input parameter of type BIGINT that specifies a specific application handle identifying the connection for which the metrics are to be returned. If the argument is null, metrics are returned for all connections

member

An input parameter of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

system_appls

An input parameter of data type SMALLINT that specifies whether information for system applications is returned. The following values are valid:

- 0 or NULL: System application information is not returned. NULL is the default if a value for *system_appls* is not provided.
- 1: User and system application information is returned.

Information returned

Table 121. Information returned for MON_GET_CONNECTION

| Column name | Data type | Description |
|-------------------------|--------------|---|
| APPLICATION_HANDLE | BIGINT | application_handle - Application handle |
| APPLICATION_NAME | VARCHAR(128) | appl_name - Application name |
| APPLICATION_ID | VARCHAR(128) | appl_id - Application ID |
| MEMBER | SMALLINT | member - Database member |
| CLIENT_WRKSTNNAME | VARCHAR(255) | client_wrkstnname - Client workstation name monitor element |
| CLIENT_ACCTNG | VARCHAR(255) | client_acctng - Client accounting string |
| CLIENT_USERID | VARCHAR(255) | client_userid - Client user ID monitor element |
| CLIENT_APPLNAME | VARCHAR(255) | client_applname - Client application name monitor element |
| CLIENT_PID | BIGINT | client_pid - Client process ID |
| CLIENT_PRDID | VARCHAR(128) | client_prdid - Client product and version ID |
| CLIENT_PLATFORM | VARCHAR(12) | client_platform - Client platform |
| CLIENT_PROTOCOL | VARCHAR(10) | client_protocol - Client communication protocol |
| SYSTEM_AUTH_ID | VARCHAR(128) | system_auth_id - System authorization identifier |
| SESSION_AUTH_ID | VARCHAR(128) | session_auth_id - Session authorization ID |
| COORD_MEMBER | SMALLINT | coord_member - Coordinating member |
| CONNECTION_START_TIME | TIMESTAMP | connection_start_time - Connection start time |
| ACT_ABORTED_TOTAL | BIGINT | act_aborted_total - Total aborted activities |
| ACT_COMPLETED_TOTAL | BIGINT | act_completed_total - Total completed activities |
| ACT_REJECTED_TOTAL | BIGINT | act_rejected_total - Total rejected activities |
| AGENT_WAIT_TIME | BIGINT | agent_wait_time - Agent wait time |
| AGENT_WAITS_TOTAL | BIGINT | agent_waits_total - Total agent waits |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|-----------------------|-----------|--|
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer pool XDA data physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer pool XDA data writes |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| CLIENT_IDLE_WAIT_TIME | BIGINT | client_idle_wait_time - Client idle wait time |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| FCM_RECV_VOLUME | BIGINT | fcm_recv_volume - FCM recv volume |
| FCM_RECVS_TOTAL | BIGINT | fcm_recvs_total - FCM recvs total |
| FCM_SEND_VOLUME | BIGINT | fcm_send_volume - FCM send volume |
| FCM_SENDS_TOTAL | BIGINT | fcm_sends_total - FCM sends total |
| FCM_RECV_WAIT_TIME | BIGINT | fcm_recv_wait_time - FCM recv wait time |
| FCM_SEND_WAIT_TIME | BIGINT | fcm_send_wait_time - FCM send wait time |
| IPC_RECV_VOLUME | BIGINT | ipc_recv_volume - Interprocess communication recv volume |
| IPC_RECV_WAIT_TIME | BIGINT | ipc_recv_wait_time - Interprocess communication recv wait time |
| IPC_RECVS_TOTAL | BIGINT | ipc_recvs_total - Interprocess communication recvs total |
| IPC_SEND_VOLUME | BIGINT | ipc_send_volume - Interprocess communication send volume |
| IPC_SEND_WAIT_TIME | BIGINT | ipc_send_wait_time - Interprocess communication send wait time |
| IPC_SENDS_TOTAL | BIGINT | ipc_sends_total - Interprocess communication send total |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOG_BUFFER_WAIT_TIME | BIGINT | log_buffer_wait_time - Log buffer wait time |
| NUM_LOG_BUFFER_FULL | BIGINT | num_log_buffer_full - Number of full log buffers |
| LOG_DISK_WAIT_TIME | BIGINT | log_disk_wait_time - Log disk wait time |
| LOG_DISK_WAITS_TOTAL | BIGINT | log_disk_waits_total - Log disk waits total |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|----------------------------------|-----------|---|
| NUM_LOCKS_HELD | BIGINT | locks_held - Locks held |
| RQSTS_COMPLETED_TOTAL | BIGINT | rqsts_completed_total - Total requests completed |
| ROWS_MODIFIED | BIGINT | rows_modified - Rows modified |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_RETURNED | BIGINT | rows_returned - Rows returned |
| TCPIP_RECV_VOLUME | BIGINT | tcpip_recv_volume - TCP/IP received volume |
| TCPIP_SEND_VOLUME | BIGINT | tcpip_send_volume - TCP/IP send volume |
| TCPIP_RECV_WAIT_TIME | BIGINT | tcpip_recv_wait_time - TCP/IP recv wait time |
| TCPIP_RECVS_TOTAL | BIGINT | tcpip_recv_total - TCP/IP recvs total |
| TCPIP_SEND_WAIT_TIME | BIGINT | tcpip_send_wait_time - TCP/IP send wait time |
| TCPIP_SENDS_TOTAL | BIGINT | tcpip_sends_total - TCP/IP sends total |
| TOTAL_APP_RQST_TIME | BIGINT | total_app_rqst_time - Total application request time |
| TOTAL_RQST_TIME | BIGINT | total_rqst_time - Total request time |
| WLM_QUEUE_TIME_TOTAL | BIGINT | wlm_queue_time_total - Workload manager total queue time |
| WLM_QUEUE_ASSIGNMENTS_TOTAL | BIGINT | wlm_queue_assignments_total - Workload manager total queue assignments |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time |
| TOTAL_WAIT_TIME | BIGINT | total_wait_time - Total wait time |
| APP_RQSTS_COMPLETED_TOTAL | BIGINT | app_rqsts_completed_total - Total application requests completed |
| TOTAL_SECTION_SORT_TIME | BIGINT | total_section_sort_time - Total section sort time |
| TOTAL_SECTION_SORT_PROC_TIME | BIGINT | total_section_sort_proc_time - Total section sort processing time |
| TOTAL_SECTION_SORTS | BIGINT | total_section_sorts - Total section sorts |
| TOTAL_SORTS | BIGINT | total_sorts - Total Sorts |
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post threshold sorts |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post shared threshold sorts |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| TOTAL_COMPILE_TIME | BIGINT | total_compile_time - Total compile time |
| TOTAL_COMPILE_PROC_TIME | BIGINT | total_compile_proc_time - Total compile processing time |
| TOTAL_COMPILATIONS | BIGINT | total_compilations - Total compilations |
| TOTAL_IMPLICIT_COMPILE_TIME | BIGINT | total_implicit_compile_time - Total implicit compile time |
| TOTAL_IMPLICIT_COMPILE_PROC_TIME | BIGINT | total_implicit_compile_proc_time - Total implicit compile processing time |
| TOTAL_IMPLICIT_COMPILATIONS | BIGINT | total_implicit_compilations - Total implicit complications |
| TOTAL_SECTION_TIME | BIGINT | total_section_time - Total section time |
| TOTAL_SECTION_PROC_TIME | BIGINT | total_section_proc_time - Total section processing time |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|------------------------------|-----------|---|
| TOTAL_APP_SECTION_EXECUTIONS | BIGINT | total_app_section_executions - Total section executions |
| TOTAL_ACT_TIME | BIGINT | total_act_time - Total activity time |
| TOTAL_ACT_WAIT_TIME | BIGINT | total_act_wait_time - Total activity wait time |
| ACT_RQSTS_TOTAL | BIGINT | act_rqsts_total - Total activity requests |
| TOTAL_ROUTINE_TIME | BIGINT | total_routine_time - Total routine time |
| TOTAL_ROUTINE_INVOCATIONS | BIGINT | total_routine_invocations - Total routine invocations |
| TOTAL_COMMIT_TIME | BIGINT | total_commit_time - Total commit time |
| TOTAL_COMMIT_PROC_TIME | BIGINT | total_commit_proc_time - Total commits processing time |
| TOTAL_APP_COMMITS | BIGINT | total_app_commits - Total application commits |
| INT_COMMITS | BIGINT | int_commits - Internal commits |
| TOTAL_ROLLBACK_TIME | BIGINT | total_rollback_time - Total rollback time |
| TOTAL_ROLLBACK_PROC_TIME | BIGINT | total_rollback_proc_time - Total rollback processing time |
| TOTAL_APP_ROLLBACKS | BIGINT | total_app_rollbacks - Total application rollbacks |
| INT_ROLLBACKS | BIGINT | int_rollbacks - Internal rollbacks |
| TOTAL_RUNSTATS_TIME | BIGINT | total_runstats_time - Total runtime statistics |
| TOTAL_RUNSTATS_PROC_TIME | BIGINT | total_runstats_proc_time - Total runtime statistics processing time |
| TOTAL_RUNSTATS | BIGINT | total_runstats - Total runtime statistics |
| TOTAL_REORG_TIME | BIGINT | total_reorg_time - Total reorganization time |
| TOTAL_REORG_PROC_TIME | BIGINT | total_reorg_proc_time - Total reorganization processing time |
| TOTAL_REORGS | BIGINT | total_reorgs - Total reorganizations |
| TOTAL_LOAD_TIME | BIGINT | total_load_time - Total load time |
| TOTAL_LOAD_PROC_TIME | BIGINT | total_load_proc_time - Total load processing time |
| TOTAL_LOADS | BIGINT | total_loads - Total loads |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups |
| THRESH_VIOLATIONS | BIGINT | thresh_violations - Number of threshold violations |
| NUM_LW_THRESH_EXCEEDED | BIGINT | num_lw_thresh_exceeded - Number of thresholds exceeded |
| LOCK_WAITS_GLOBAL | BIGINT | lock_waits_global - Lock waits global |
| LOCK_WAIT_TIME_GLOBAL | BIGINT | lock_wait_time_global - Lock wait time global |
| LOCK_TIMEOUTS_GLOBAL | BIGINT | lock_timeouts_global - Lock timeouts global |
| LOCK_ESCALS_MAXLOCKS | BIGINT | lock_escal_maxlocks - Number of maxlocks lock escalations |
| LOCK_ESCALS_LOCKLIST | BIGINT | lock_escal_locklist - Number of locklist lock escalations |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|--------------------------------|--------------|--|
| LOCK_ESCALS_GLOBAL | BIGINT | lock_escals_global - Number of global lock escalations |
| RECLAIM_WAIT_TIME | BIGINT | reclaim_wait_time - Reclaim wait time |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME | BIGINT | spacemappage_reclaim_wait_time - Space map page reclaim wait time |
| CF_WAITS | BIGINT | cf_waits - Number of cluster caching facility waits |
| CF_WAIT_TIME | BIGINT | cf_wait_time - cluster caching facility wait time |
| POOL_DATA_GBP_L_READS | BIGINT | pool_data_gbp_l_reads - Group buffer pool data logical reads |
| POOL_DATA_GBP_P_READS | BIGINT | pool_data_gbp_p_reads - Group buffer pool data physical reads |
| POOL_DATA_LBP_PAGES_FOUND | BIGINT | pool_data_lbp_pages_found - Local buffer pool found data pages |
| POOL_DATA_GBP_INVALID_PAGES | BIGINT | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages |
| POOL_INDEX_GBP_L_READS | BIGINT | pool_index_gbp_l_reads - Group buffer pool index logical reads |
| POOL_INDEX_GBP_P_READS | BIGINT | pool_index_gbp_p_reads - Group buffer pool index physical reads |
| POOL_INDEX_LBP_PAGES_FOUND | BIGINT | pool_index_lbp_pages_found - Local buffer pool index pages found |
| POOL_INDEX_GBP_INVALID_PAGES | BIGINT | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages |
| POOL_XDA_GBP_L_READS | BIGINT | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |
| POOL_XDA_GBP_P_READS | BIGINT | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |
| POOL_XDA_LBP_PAGES_FOUND | BIGINT | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |
| POOL_XDA_GBP_INVALID_PAGES | BIGINT | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| AUDIT_EVENTS_TOTAL | BIGINT | audit_events_total - Total audit events |
| AUDIT_FILE_WRITES_TOTAL | BIGINT | audit_file_writes_total - Total Audit files written |
| AUDIT_FILE_WRITE_WAIT_TIME | BIGINT | audit_file_write_wait_time - Audit file write wait time |
| AUDIT_SUBSYSTEM_WAITS_TOTAL | BIGINT | audit_subsystem_waits_total - Total audit subsystem waits |
| AUDIT_SUBSYSTEM_WAIT_TIME | BIGINT | audit_subsystem_wait_time - Audit subsystem wait time |
| CLIENT_HOSTNAME | VARCHAR(255) | client_hostname - Client hostname monitor element |
| CLIENT_PORT_NUMBER | INTEGER | client_port_number - Client port number monitor element |
| DIAGLOG_WRITES_TOTAL | BIGINT | diaglog_writes_total - Diag log total writes |
| DIAGLOG_WRITE_WAIT_TIME | BIGINT | diaglog_write_wait_time - Diag log write time |
| FCM_MESSAGE_RECVS_TOTAL | BIGINT | fcm_message_recvs_total - FCM message recvs total |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|-----------------------------------|-----------------------------|--|
| FCM_MESSAGE_RECV_VOLUME | BIGINT | fcm_message_recv_volume - FCM message recv volume |
| FCM_MESSAGE_RECV_WAIT_TIME | BIGINT | fcm_message_recv_wait_time - FCM message recv wait time |
| FCM_MESSAGE_SENDS_TOTAL | BIGINT | fcm_message_sends_total - FCM message sends total |
| FCM_MESSAGE_SEND_VOLUME | BIGINT | fcm_message_send_volume - FCM message send volume |
| FCM_MESSAGE_SEND_WAIT_TIME | BIGINT | fcm_message_send_wait_time - FCM message send wait time |
| FCM_TQ_RECVS_TOTAL | BIGINT | fcm_tq_recvs_total - FCM tablequeue recvs total |
| FCM_TQ_RECV_VOLUME | BIGINT | fcm_tq_recv_volume - FCM tablequeue recv volume |
| FCM_TQ_RECV_WAIT_TIME | BIGINT | fcm_tq_recv_wait_time - FCM tablequeue recv wait time |
| FCM_TQ_SENDS_TOTAL | BIGINT | fcm_tq_sends_total - FCM tablequeue send total |
| FCM_TQ_SEND_VOLUME | BIGINT | fcm_tq_send_volume - FCM tablequeue send volume |
| FCM_TQ_SEND_WAIT_TIME | BIGINT | fcm_tq_send_wait_time - FCM tablequeue send wait time |
| LAST_EXECUTABLE_ID | VARCHAR(32)
FOR BIT DATA | last_executable_id - Last executable identifier |
| LAST_REQUEST_TYPE | VARCHAR(32) | last_request_type - Last request type |
| TOTAL_ROUTINE_USER_CODE_PROC_TIME | BIGINT | total_routine_user_code_proc_time - Total routine user code processing time |
| TOTAL_ROUTINE_USER_CODE_TIME | BIGINT | total_routine_user_code_time - Total routine user code time |
| TQ_TOT_SEND_SPILLS | BIGINT | tq_tot_send_spills - Total number of table queue buffers overflowed |
| EVMON_WAIT_TIME | BIGINT | evmon_wait_time - Event monitor wait time |
| EVMON_WAITS_TOTAL | BIGINT | evmon_waits_total - Event monitor total waits |
| TOTAL_EXTENDED_LATCH_WAIT_TIME | BIGINT | total_extended_latch_wait_time - Total extended latch wait time |
| TOTAL_EXTENDED_LATCH_WAITS | BIGINT | total_extended_latch_waits - Total extended latch waits |
| INTRA_PARALLEL_STATE | VARCHAR(3) | intra_parallel_state - Currentstate of intrapartition parallelism |
| TOTAL_STATS_FABRICATION_TIME | BIGINT | total_stats_fabrication_time - Total statistics fabrication time |
| TOTAL_STATS_FABRICATION_PROC_TIME | BIGINT | total_stats_fabrication_proc_time - Total statistics fabrication processing time |
| TOTAL_STATS_FABRICATIONS | BIGINT | total_stats_fabrications - Total statistics fabrications |
| TOTAL_SYNC_RUNSTATS_TIME | BIGINT | total_sync_runstats_time - Total synchronous RUNSTATS time |
| TOTAL_SYNC_RUNSTATS_PROC_TIME | BIGINT | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|------------------------------------|-----------|--|
| TOTAL_SYNC_RUNSTATS | BIGINT | total_sync_runstats - Total synchronous RUNSTATS activities |
| TOTAL_DISP_RUN_QUEUE_TIME | BIGINT | total_disp_run_queue_time - Total dispatcher run queue time |
| TOTAL_PEDS | BIGINT | total_peds - Total partial early distincts |
| DISABLED_PEDS | BIGINT | disabled_peds - Disabled partial early distincts |
| POST_THRESHOLD_PEDS | BIGINT | post_threshold_peds - Partial early distincts threshold |
| TOTAL_PEAS | BIGINT | total_peas - Total partial early aggregations |
| POST_THRESHOLD_PEAS | BIGINT | post_threshold_peas - Partial early aggregation threshold |
| TQ_SORT_HEAP_REQUESTS | BIGINT | tq_sort_heap_requests - Table queue sort heap requests |
| TQ_SORT_HEAP_REJECTIONS | BIGINT | tq_sort_heap_rejections - Table queue sort heap rejections |
| POOL_QUEUED_ASYNC_DATA_REQS | BIGINT | pool_queued_async_data_reqs - Data prefetch requests |
| POOL_QUEUED_ASYNC_INDEX_REQS | BIGINT | pool_queued_async_index_reqs - Index prefetch requests |
| POOL_QUEUED_ASYNC_XDA_REQS | BIGINT | pool_queued_async_xda_reqs - XDA prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_OTHER_REQS | BIGINT | pool_queued_async_other_reqs - Non-prefetch requests |
| POOL_QUEUED_ASYNC_DATA_PAGES | BIGINT | pool_queued_async_data_pages - Data pages prefetch requests |
| POOL_QUEUED_ASYNC_INDEX_PAGES | BIGINT | pool_queued_async_index_pages - Index pages prefetch requests |
| POOL_QUEUED_ASYNC_XDA_PAGES | BIGINT | pool_queued_async_xda_pages - XDA pages prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES | BIGINT | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES | BIGINT | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_DATA_REQS | BIGINT | pool_failed_async_data_reqs - Failed data prefetch requests |
| POOL_FAILED_ASYNC_INDEX_REQS | BIGINT | pool_failed_async_index_reqs - Failed index prefetch requests |
| POOL_FAILED_ASYNC_XDA_REQS | BIGINT | pool_failed_async_xda_reqs - Failed XDA prefetch requests |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|---|-----------|--|
| POOL_FAILED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_OTHER_REQS | BIGINT | pool_failed_async_other_reqs - Failed non-prefetch requests |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| PREFETCH_WAITS | BIGINT | prefetch_waits - Prefetcher wait count |
| APP_ACT_COMPLETED_TOTAL | BIGINT | app_act_completed_total - Total successful external coordinator activities |
| APP_ACT_ABORTED_TOTAL | BIGINT | app_act_aborted_total - Total failed external coordinator activities |
| APP_ACT_REJECTED_TOTAL | BIGINT | app_act_rejected_total - Total rejected external coordinator activities |
| TOTAL_CONNECT_REQUEST_TIME | BIGINT | total_connect_request_time - Total connection or switch user request time |
| TOTAL_CONNECT_REQUEST_PROC_TIME | BIGINT | total_connect_request_proc_time - Total connection or switch user request processing time |
| TOTAL_CONNECT_REQUESTS | BIGINT | total_connect_requests - Connection or switch user requests |
| TOTAL_CONNECT_AUTHENTICATION_TIME | BIGINT | total_connect_authentication_time - Total connection or switch user authentication request time |
| TOTAL_CONNECT_AUTHENTICATION_PROC_TIME | BIGINT | total_connect_authentication_proc_time - Total connection authentication processing time |
| TOTAL_CONNECT_AUTHENTIFICATIONS | BIGINT | total_connect_authentications - Connections or switch user authentications performed |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |
| COMM_EXIT_WAIT_TIME | BIGINT | comm_exit_wait_time - Communication buffer exit wait time monitor element |
| COMM_EXIT_WAITS | BIGINT | comm_exit_waits - Communication buffer exit number of waits monitor element |
| IDA_SEND_WAIT_TIME | BIGINT | ida_send_wait_time - Time spent waiting to send data monitor element |
| IDA_SENDS_TOTAL | BIGINT | ida_sends_total - Number of times data sent monitor element |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|------------------------------|-----------|---|
| IDA_SEND_VOLUME | BIGINT | ida_send_volume - Total data volume sent monitor element |
| IDA_RECV_WAIT_TIME | BIGINT | ida_recv_wait_time - Time spent waiting to receive data monitor element |
| IDA_RECVS_TOTAL | BIGINT | ida_recv_total - Number of times data received monitor element |
| IDA_RECV_VOLUME | BIGINT | ida_recv_volume - Total data volume received monitor element |
| MEMBER_SUBSET_ID | INTEGER | member_subset_id - Member subset ID |
| IS_SYSTEM_APPL | SMALLINT | is_system_appl - Is System Application monitor element |
| LOCK_TIMEOUT_VAL | BIGINT | lock_timeout_val - Lock timeout value monitor element |
| CURRENT_ISOLATION | CHAR(2) | current_isolation - Current isolation level |
| NUM_LOCKS_WAITING | BIGINT | locks_waiting - Current agents waiting on locks monitor element |
| UOW_CLIENT_IDLE_WAIT_TIME | BIGINT | uow_client_idle_wait_time - Client idle time within a unit of work |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted monitor element |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted monitor element |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated monitor element |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total Hash Joins monitor element |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total Hash Loops monitor element |
| HASH_JOIN_OVERFLOWS | BIGINT | hash_join_overflows - Hash Join Overflows monitor element |
| HASH_JOIN_SMALL_OVERFLOWS | BIGINT | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| POST_SHRTHRESHOLD_HASH_JOINS | BIGINT | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP Functions monitor element |
| OLAP_FUNC_OVERFLOWS | BIGINT | olap_func_overflows - OLAP Function Overflows monitor element |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element |
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL Statements Attempted monitor element |
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed Statement Operations monitor element |
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL Statements Executed monitor element |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|---------------------------------------|-----------|---|
| MERGE_SQL_STMTS | BIGINT | merge_sql_stmts - Merge SQL statements executed |
| XQUERY_STMTS | BIGINT | xquery_stmts - XQuery Statements Attempted monitor element |
| IMPLICIT_REBINDS | BIGINT | implicit_rebinds - number of implicit rebinds |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/Precompiles Attempted monitor element |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal Rows Inserted monitor element |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal Rows Updated monitor element |
| CALL_SQL_STMTS | BIGINT | call_sql_stmts - CALL SQL statements executed |
| POOL_COL_L_READS | BIGINT | pool_col_l_reads - Buffer pool column-organized logical reads |
| POOL_TEMP_COL_L_READS | BIGINT | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| POOL_COL_P_READS | BIGINT | pool_col_p_reads - Buffer pool column-organized physical reads |
| POOL_TEMP_COL_P_READS | BIGINT | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| POOL_COL_LBP_PAGES_FOUND | BIGINT | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found |
| POOL_COL_WRITES | BIGINT | pool_col_writes - Buffer pool column-organized writes |
| POOL_COL_GBP_L_READS | BIGINT | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| POOL_COL_GBP_P_READS | BIGINT | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| POOL_COL_GBP_INVALID_PAGES | BIGINT | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |
| POOL_QUEUED_ASYNC_COL_REQS | BIGINT | pool_queued_async_col_reqs - Column-organized prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_REQS | BIGINT | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| POOL_QUEUED_ASYNC_COL_PAGES | BIGINT | pool_queued_async_col_pages - Column-organized page prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_PAGES | BIGINT | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| POOL_FAILED_ASYNC_COL_REQS | BIGINT | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| POOL_FAILED_ASYNC_TEMP_COL_REQS | BIGINT | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|--------------------------------------|--------------|---|
| TOTAL_COL_TIME | BIGINT | total_col_time - Total column-organized time |
| TOTAL_COL_PROC_TIME | BIGINT | total_col_proc_time - Total column-organized processing time |
| TOTAL_COL_EXECUTIONS | BIGINT | total_col_executions - Total column-organized executions |
| CLIENT_IPADDR | VARCHAR(128) | client_ipaddr - Client IP address monitor element |
| SQL_REQS_SINCE_COMMIT | BIGINT | sql_reqs_since_commit - SQL Requests Since Last Commit monitor element |
| UOW_START_TIME | TIMESTAMP | uow_start_time - Unit of work start timestamp monitor element |
| UOW_STOP_TIME | TIMESTAMP | uow_stop_time - Unit of work stop timestamp monitor element |
| PREV_UOW_STOP_TIME | TIMESTAMP | prev_uow_stop_time - Previous Unit of Work Completion Timestamp monitor element |
| UOW_COMP_STATUS | VARCHAR(14) | uow_comp_status - Unit of Work Completion Status monitor element |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of Associated Agents monitor element |
| ASSOCIATED_AGENTS_TOP | BIGINT | associated_agents_top - Maximum Number of Associated Agents monitor element |
| WORKLOAD_OCCURRENCE_STATE | VARCHAR(32) | workload_occurrence_state - Workload occurrence state monitor element |
| POST_THRESHOLD_HASH_JOINS | BIGINT | post_threshold_hash_joins - Hash Join Threshold monitor element |
| POOL_DATA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|---|--------------|--|
| POOL_COL_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_READ_TIME | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_WRITE_TIME | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| TOTAL_HASH_GRPBYS | BIGINT | total_hash_grpbys - Total hash group by operations |
| HASH_GRPBY_OVERFLOWES | BIGINT | hash_grpby_overflows - Hash group by overflows |
| POST_THRESHOLD_HASH_GRPBYS | BIGINT | post_threshold_hash_grpbys - Hash group by threshold |
| EXECUTION_ID | VARCHAR(128) | execution_id - User Login ID monitor element |
| POST_THRESHOLD_OLAP_FUNCS | BIGINT | post_threshold_olap_funcs - OLAP function threshold monitor element |
| POST_THRESHOLD_COL_VECTOR_CONSUMERS | BIGINT | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| TOTAL_COL_VECTOR_CONSUMERS | BIGINT | total_col_vector_consumers - Total columnar_vector consumers |
| ACTIVE_HASH_GRPBYS | BIGINT | active_hash_grpbys - Active hash GROUP BY operations |
| ACTIVE_HASH_JOINS | BIGINT | active_hash_joins - Active hash joins |
| ACTIVE_OLAP_FUNCS | BIGINT | active_olap_funcs - Active OLAP Functions |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|------------------------------|-----------|--|
| ACTIVE_PEAS | BIGINT | active_peas - Active partial early aggregations operations |
| ACTIVE_PEDS | BIGINT | active_peds - Active partial early distinct operations |
| ACTIVE_SORT_CONSUMERS | BIGINT | active_sort_consumers - Active sort consumers |
| ACTIVE_SORTS | BIGINT | active_sorts - Active Sorts |
| ACTIVE_COL_VECTOR_CONSUMERS | BIGINT | active_col_vector_consumers - Active columnar_vector consumers |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total Sort Heap Allocated |
| SORT_SHRHEAP_ALLOCATED | BIGINT | sort_shrheap_allocated - Sort Share Heap Currently Allocated |
| TOTAL_BACKUP_TIME | BIGINT | total_backup_time - Total elapsed time for online backups |
| TOTAL_BACKUP_PROC_TIME | BIGINT | total_backup_proc_time - Total non-wait time for online backups |
| TOTAL_BACKUPS | BIGINT | total_backups - Total backups |
| TOTAL_INDEX_BUILD_TIME | BIGINT | total_index_build_time - Total time spent building indexes due to index creation, recreation or rebuilding |
| TOTAL_INDEX_BUILD_PROC_TIME | BIGINT | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation, recreation or rebuilding |
| TOTAL_INDEXES_BUILT | BIGINT | total_indexes_built - Total number of indexes built |
| EXT_TABLE_RECV_WAIT_TIME | BIGINT | ext_table_recv_wait_time - Total agent wait time for external table readers monitor element |
| EXT_TABLE_RECVS_TOTAL | BIGINT | ext_table_recvs_total - Total row batches received from external table readers monitor element |
| EXT_TABLE_RECV_VOLUME | BIGINT | ext_table_recv_volume - Total data received from external table readers monitor element |
| EXT_TABLE_READ_VOLUME | BIGINT | ext_table_read_volume - Total data read by external table readers monitor element |
| EXT_TABLE_SEND_WAIT_TIME | BIGINT | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| EXT_TABLE_SENDS_TOTAL | BIGINT | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| EXT_TABLE_SEND_VOLUME | BIGINT | ext_table_send_volume - Total data sent to external table writers monitor element |
| EXT_TABLE_WRITE_VOLUME | BIGINT | ext_table_write_volume - Total data written by external table writers monitor element |
| FCM_TQ_RECV_WAITS_TOTAL | BIGINT | fcm_tq_recv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| FCM_MESSAGE_RECV_WAITS_TOTAL | BIGINT | fcm_message_recv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| FCM_TQ_SEND_WAITS_TOTAL | BIGINT | fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |

Table 121. Information returned for MON_GET_CONNECTION (continued)

| Column name | Data type | Description |
|-------------------------------|--------------|--|
| FCM_MESSAGE_SEND_WAITS_TOTAL | BIGINT | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| FCM_SEND_WAITS_TOTAL | BIGINT | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| FCM_RECV_WAITS_TOTAL | BIGINT | fcm_recv_waits_total - Number of times spent waiting to receive data through FCM monitor element |
| COL_VECTOR_CONSUMER_OVERFLOW | BIGINT | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |
| TOTAL_COL_SYNOPSIS_TIME | BIGINT | total_col_synopsis_time - Total column-organized synopsis table time monitor element |
| TOTAL_COL_SYNOPSIS_PROC_TIME | BIGINT | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| TOTAL_COL_SYNOPSIS_EXECUTIONS | BIGINT | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |
| COL_SYNOPSIS_ROWS_INSERTED | BIGINT | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| LOB_PREFETCH_WAIT_TIME | BIGINT | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| LOB_PREFETCH_REQS | BIGINT | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| FED_ROWS_DELETED | BIGINT | fed_rows_deleted - Rows deleted by a federation system monitor element |
| FED_ROWS_INSERTED | BIGINT | fed_rows_inserted - Rows inserted by a federation system monitor element |
| FED_ROWS_UPDATED | BIGINT | fed_rows_updated - Rows updated by a federation system monitor element |
| FED_ROWS_READ | BIGINT | fed_rows_read - Rows read by a federation system monitor element |
| FED_WAIT_TIME | BIGINT | fed_wait_time - Time spent by a federation server monitor element |
| FED_WAITS_TOTAL | BIGINT | fed_waits_total - Total number of execution times for a federation server monitor element |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Insert count for SQL workspace monitor element |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section lookups - SQL workspace access count monitor element |
| CONNECTION_REUSABILITY_STATUS | SMALLINT | connection_reusability_status - Connection reusability status monitor element |
| REUSABILITY_STATUS_REASON | VARCHAR(255) | reusability_status_reason - Reusability status reason monitor element |
| ADM_OVERFLOW | BIGINT | Reserved for future use. |
| ADM_BYPASS_ACT_TOTAL | BIGINT | Reserved for future use. |

Usage notes

The metrics returned by the `MON_GET_CONNECTION` table function represent the accumulation of all metrics for requests that were submitted by a connection. Metrics are rolled up at unit of work boundaries, and periodically during the execution of requests. Therefore, the values reported by this table function reflect the current state of the system at the time of the most recent rollup. Metrics are strictly increasing in value. To determine the value of a given metric for an interval of time, use the `MON_GET_CONNECTION` table function to query the metric at the start and end of the interval, and compute the difference.

Request metrics are controlled through the `COLLECT REQUEST METRICS` clause on service superclasses and the `mon_req_metrics` database configuration parameter at the database level. Metrics are only collected for a request if the request is processed by an agent in a service subclass whose parent service superclass has request metrics enabled, or if request metrics collection is enabled for the entire database. By default, request metrics are enabled at the database level. If request metrics are disabled at the database level and for a service superclass, then the metrics reported for each connection that is mapped to that service superclass will stop increasing (or remain at 0 if request metrics were disabled at database activation time).

Tip: As a connection can be mapped to more than one service superclass during its lifetime, the metrics reported through the `MON_GET_CONNECTION` table function might represent a subset of the metrics for all requests submitted over the connection. This might occur if metrics collection is disabled for some of the superclasses that are mapped by the connection.

The `MON_GET_CONNECTION` table function returns one row of data per connection and per member. No aggregation across members (for a service class or more), is performed. However, aggregation can be achieved through SQL queries.

Example

Display connections that return the highest volume of data to clients, ordered by rows returned.

```
SELECT application_handle,  
       rows_returned,  
       tcpip_send_volume  
FROM TABLE(MON_GET_CONNECTION(cast(NULL as bigint), -2)) AS t  
ORDER BY rows_returned DESC
```

The following is an example of output from this query.

| APPLICATION_HANDLE | ROWS_RETURNED | TCPIP_SEND_VOLUME |
|--------------------|---------------|-------------------|
| ----- | ----- | ----- |
| | 55 | 6 |
| | | 0 |

1 record(s) selected.

MON_GET_CONNECTION_DETAILS table function - Get connection metrics as an XML document

The `MON_GET_CONNECTION_DETAILS` table function returns metrics for one or more connections.

The metrics are returned in an XML document.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

```
►► MON_GET_CONNECTION_DETAILS (—application_handle—, —member—, —system_appls—)
```

The schema is SYSPROC.

Routine parameters

application_handle

An input argument of type BIGINT that specifies a specific application handle identifying the connection for which the metrics are to be returned. If the argument is NULL, metrics are returned for all connections.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

system_appls

An input parameter of data type SMALLINT that specifies whether information for system applications is returned. The following values are valid:

- 0 or NULL: System application information is not returned. NULL is the default if a value for *system_appls* is not provided.
- 1: User and system application information is returned.

Information returned

Table 122. Information returned for MON_GET_CONNECTION_DETAILS

| Column Name | Data Type | Description |
|--------------------|-----------|--|
| APPLICATION_HANDLE | BIGINT | application_handle - Application handle |
| MEMBER | SMALLINT | member - Database member |
| DETAILS | BLOB(1M) | XML document containing detailed metrics for the unit of work. See Table 123 on page 605 for a description of the elements in this document. |

The following example shows the structure of the XML document that is returned in the DETAILS column.

```

<db2_connection xmlns="http://www.ibm.com/xmlns/prod/db2/mon" release="90700000">
  <application_handle>21</application_handle>
  <member>0</member>
  <system_metrics release="90700000">
    <act_aborted_total>5</act_aborted_total>
    ...
    <wlm_queue_assignments_total>3</wlm_queue_assignments_total>
  </system_metrics>
</db2_connection>

```

For the full schema, see `sqllib/misc/DB2MonRoutines.xsd`.

This document uses the following non-primitive XML type definitions:

```

<xs:simpleType name="db2DbObjectString">
  <xs:restriction base="xs:string">
    <xs:maxLength value="128"/>
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="db2PartitionNum">
  <xs:restriction base="xs:nonNegativeInteger">
    <xs:maxInclusive value="999"/>
  </xs:restriction>
</xs:simpleType>

```

Table 123. Detailed metrics returned for MON_GET_CONNECTION_DETAILS

| Element Name | Data Type | Description |
|-----------------------------|-----------------------|--|
| act_aborted_total | xs:nonNegativeInteger | act_aborted_total - Total aborted activities |
| act_completed_total | xs:nonNegativeInteger | act_completed_total - Total completed activities |
| act_rejected_total | xs:nonNegativeInteger | act_rejected_total - Total rejected activities |
| act_rqsts_total | xs:nonNegativeInteger | act_rqsts_total - Total activity requests |
| active_col_vector_consumers | xs:long | active_col_vector_consumers - Active columnar_vector consumers |
| active_hash_grpbys | xs:long | active_hash_grpbys - Active hash GROUP BY operations |
| active_hash_joins | xs:long | active_hash_joins - Active hash joins |
| active_olap_funcs | xs:long | active_olap_funcs - Active OLAP Functions |
| active_peas | xs:long | active_peas - Active partial early aggregations operations |
| active_peds | xs:long | active_peds - Active partial early distinct operations |
| active_sort_consumers | xs:long | active_sort_consumers - Active sort consumers |
| active_sorts | xs:long | active_sorts - Active Sorts |
| adm_bypass_act_total | xs:long | Reserved for future use. |
| adm_overflows | xs:long | Reserved for future use. |
| agent_id | xs:nonNegativeInteger | agent_id - Application handle (agent ID) monitor element |
| agent_wait_time | xs:nonNegativeInteger | agent_wait_time - Agent wait time |
| agent_waits_total | xs:nonNegativeInteger | agent_waits_total - Total agent waits |
| app_act_aborted_total | xs:nonNegativeInteger | app_act_aborted_total - Total failed external coordinator activities |
| app_act_completed_total | xs:nonNegativeInteger | app_act_completed_total - Total successful external coordinator activities |
| app_act_rejected_total | xs:nonNegativeInteger | app_act_rejected_total - Total rejected external coordinator activities |
| app_rqsts_completed_total | xs:nonNegativeInteger | app_rqsts_completed_total - Total application requests completed |
| appl_section_inserts | xs:nonNegativeInteger | appl_section_inserts - Insert count for SQL workspace monitor element |

Table 123. Detailed metrics returned for MON_GET_CONNECTION_DETAILS (continued)

| Element Name | Data Type | Description |
|-------------------------------|-----------------------|--|
| appl_section_lookups | xs:nonNegativeInteger | appl_section_lookups - SQL workspace access count monitor element |
| application_handle | xs:nonNegativeInteger | application_handle - Application handle |
| application_id | xs:string | appl_id - Application ID |
| application_name | xs:string | appl_name - Application name |
| associated_agents_top | xs:long | associated_agents_top - Maximum Number of Associated Agents monitor element |
| audit_events_total | xs:nonNegativeInteger | audit_events_total - Total audit events |
| audit_file_write_wait_time | xs:nonNegativeInteger | audit_file_write_wait_time - Audit file write wait time |
| audit_file_writes_total | xs:nonNegativeInteger | audit_file_writes_total - Total Audit files written |
| audit_subsystem_wait_time | xs:nonNegativeInteger | audit_subsystem_wait_time - Audit subsystem wait time |
| audit_subsystem_waits_total | xs:nonNegativeInteger | audit_subsystem_waits_total - Total audit subsystem waits |
| binds_precompiles | xs:long | binds_precompiles - Binds/Precompiles Attempted monitor element |
| call_sql_stmts | xs:long | call_sql_stmts - CALL SQL statements executed |
| cat_cache_inserts | xs:nonNegativeInteger | cat_cache_inserts - Catalog cache inserts |
| cat_cache_lookups | xs:nonNegativeInteger | cat_cache_lookups - Catalog cache lookups |
| client_acctng | xs:string (255) | client_acctng - Client accounting string |
| client_applname | xs:string (255) | client_applname - Client application name |
| client_hostname | xs:string | client_hostname - Client hostname |
| client_idle_wait_time | xs:nonNegativeInteger | client_idle_wait_time - Client idle wait time |
| client_ipaddr | xs:string | client_ipaddr - Client IP address monitor element |
| client_pid | xs:nonNegativeInteger | client_pid - Client process ID |
| client_platform | xs:string | client_platform - Client platform |
| client_port_number | xs:nonNegativeInteger | client_port_number - Client port number |
| client_prdid | xs:string | client_prdid - Client product and version ID |
| client_protocol | xs:string | client_protocol - Client communication protocol |
| client_userid | xs:string (255) | client_userid - Client user ID |
| client_wrkstnname | xs:string (255) | client_wrkstnname - Client workstation name |
| col_synopsis_rows_inserted | xs:nonNegativeInteger | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| col_vector_consumer_overflows | xs:long | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |
| comm_exit_wait_time | xs:nonNegativeInteger | comm_exit_wait_time - Communication buffer exit wait time monitor element |
| comm_exit_waits | xs:nonNegativeInteger | comm_exit_waits - Communication buffer exit number of waits monitor element |
| connection_reusability_status | xs:short | connection_reusability_status - Connection reusability status monitor element |
| connection_start_time | xs:dateTime | connection_start_time - Connection start time |
| coord_member | xs:short | coord_member - Coordinating member |
| current_isolation | xs:string | current_isolation - Current isolation level |
| ddl_sql_stmts | xs:long | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| deadlocks | xs:nonNegativeInteger | deadlocks - Deadlocks detected |
| diaglog_write_wait_time | xs:nonNegativeInteger | diaglog_write_wait_time - Diag log write time |
| diaglog_writes_total | xs:nonNegativeInteger | diaglog_writes_total - Diag log total writes |
| direct_read_reqs | xs:nonNegativeInteger | direct_read_reqs - Direct read requests |

Table 123. Detailed metrics returned for MON_GET_CONNECTION_DETAILS (continued)

| Element Name | Data Type | Description |
|------------------------------|-----------------------|--|
| direct_read_time | xs:nonNegativeInteger | direct_read_time - Direct read time |
| direct_reads | xs:nonNegativeInteger | direct_reads - Direct reads from database |
| direct_write_reqs | xs:nonNegativeInteger | direct_write_reqs - Direct write requests |
| direct_write_time | xs:nonNegativeInteger | direct_write_time - Direct write time |
| direct_writes | xs:nonNegativeInteger | direct_writes - Direct writes to database |
| disabled_peds | xs:long | disabled_peds - Disabled partial early distincts |
| dynamic_sql_stmts | xs:long | dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element |
| evmon_wait_time | xs:nonNegativeInteger | evmon_wait_time - Event monitor wait time |
| evmon_waits_total | xs:nonNegativeInteger | evmon_waits_total - Event monitor total waits |
| execution_id | xs:string(128) | execution_id - User Login ID monitor element |
| ext_table_read_volume | xs:nonNegativeInteger | ext_table_read_volume - Total data read by external table readers monitor element |
| ext_table_rcv_volume | xs:nonNegativeInteger | ext_table_rcv_volume - Total data received from external table readers monitor element |
| ext_table_rcv_wait_time | xs:nonNegativeInteger | ext_table_rcv_wait_time - Total agent wait time for external table readers monitor element |
| ext_table_rcvs_total | xs:nonNegativeInteger | ext_table_rcvs_total - Total row batches received from external table readers monitor element |
| ext_table_send_volume | xs:nonNegativeInteger | ext_table_send_volume - Total data sent to external table writers monitor element |
| ext_table_send_wait_time | xs:nonNegativeInteger | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| ext_table_sends_total | xs:nonNegativeInteger | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| ext_table_write_volume | xs:nonNegativeInteger | ext_table_write_volume - Total data written by external table writers monitor element |
| failed_sql_stmts | xs:long | failed_sql_stmts - Failed Statement Operations monitor element |
| fcm_message_rcv_volume | xs:nonNegativeInteger | fcm_message_rcv_volume - FCM message rcv volume |
| fcm_message_rcv_wait_time | xs:nonNegativeInteger | fcm_message_rcv_wait_time - FCM message rcv wait time |
| fcm_message_rcv_waits_total | xs:nonNegativeInteger | fcm_message_rcv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| fcm_message_rcvs_total | xs:nonNegativeInteger | fcm_message_rcvs_total - FCM message rcvs total |
| fcm_message_send_volume | xs:nonNegativeInteger | fcm_message_send_volume - FCM message send volume |
| fcm_message_send_wait_time | xs:nonNegativeInteger | fcm_message_send_wait_time - FCM message send wait time |
| fcm_message_send_waits_total | xs:nonNegativeInteger | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| fcm_message_sends_total | xs:nonNegativeInteger | fcm_message_sends_total - FCM message sends total |
| fcm_rcv_volume | xs:nonNegativeInteger | fcm_rcv_volume - FCM rcv volume |
| fcm_rcv_wait_time | xs:nonNegativeInteger | fcm_rcv_wait_time - FCM rcv wait time |
| fcm_rcv_waits_total | xs:nonNegativeInteger | fcm_rcv_waits_total - Number of times spent waiting to receive data through FCM monitor element |
| fcm_rcvs_total | xs:nonNegativeInteger | fcm_rcvs_total - FCM rcvs total |
| fcm_send_volume | xs:nonNegativeInteger | fcm_send_volume - FCM send volume |
| fcm_send_wait_time | xs:nonNegativeInteger | fcm_send_wait_time - FCM send wait time |
| fcm_send_waits_total | xs:nonNegativeInteger | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| fcm_sends_total | xs:nonNegativeInteger | fcm_sends_total - FCM sends total |

Table 123. Detailed metrics returned for MON_GET_CONNECTION_DETAILS (continued)

| Element Name | Data Type | Description |
|---------------------------|-----------------------|---|
| fcmtq_recv_volume | xs:nonNegativeInteger | fcmtq_recv_volume - FCM tablequeue recv volume |
| fcmtq_recv_wait_time | xs:nonNegativeInteger | fcmtq_recv_wait_time - FCM tablequeue recv wait time |
| fcmtq_recv_waits_total | xs:nonNegativeInteger | fcmtq_recv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| fcmtq_recvs_total | xs:nonNegativeInteger | fcmtq_recvs_total - FCM tablequeue recvs total |
| fcmtq_send_volume | xs:nonNegativeInteger | fcmtq_send_volume - FCM tablequeue send volume |
| fcmtq_send_wait_time | xs:nonNegativeInteger | fcmtq_send_wait_time - FCM tablequeue send wait time |
| fcmtq_send_waits_total | xs:nonNegativeInteger | fcmtq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |
| fcmtq_sends_total | xs:nonNegativeInteger | fcmtq_sends_total - FCM tablequeue send total |
| fed_rows_deleted | xs:nonNegativeInteger | fed_rows_deleted - Rows deleted by a federation system monitor element |
| fed_rows_inserted | xs:nonNegativeInteger | fed_rows_inserted - Rows inserted by a federation system monitor element |
| fed_rows_read | xs:nonNegativeInteger | fed_rows_read - Rows read by a federation system monitor element |
| fed_rows_updated | xs:nonNegativeInteger | fed_rows_updated - Rows updated by a federation system monitor element |
| fed_wait_time | xs:nonNegativeInteger | fed_wait_time - Time spent by a federation server monitor element |
| fed_waits_total | xs:nonNegativeInteger | fed_waits_total - Total number of execution times for a federation server monitor element |
| hash_grpby_overflows | xs:long | hash_grpby_overflows - Hash group by overflows |
| hash_join_overflows | xs:long | hash_join_overflows - Hash Join Overflows monitor element |
| hash_join_small_overflows | xs:long | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| ida_recv_volume | xs:long | ida_recv_volume - Total data volume received monitor element |
| ida_recv_wait_time | xs:long | ida_recv_wait_time - Time spent waiting to receive data monitor element |
| ida_recvs_total | xs:long | ida_recvs_total - Number of times data received monitor element |
| ida_send_volume | xs:long | ida_send_volume - Total data volume sent monitor element |
| ida_send_wait_time | xs:long | ida_send_wait_time - Time spent waiting to send data monitor element |
| ida_sends_total | xs:long | ida_sends_total - Number of times data sent monitor element |
| implicit_rebinds | xs:long | implicit_rebinds - number of implicit rebinds |
| int_commits | xs:nonNegativeInteger | int_commits - Internal commits |
| int_rollback | xs:nonNegativeInteger | int_rollback - Internal rollbacks |
| int_rows_deleted | xs:long | int_rows_deleted - Internal Rows Deleted monitor element |
| int_rows_inserted | xs:long | int_rows_inserted - Internal Rows Inserted monitor element |
| int_rows_updated | xs:long | int_rows_updated - Internal Rows Updated monitor element |
| intra_parallel_state | xs:string | intra_parallel_state - Current state of intrapartition parallelism monitor element |
| ipc_recv_volume | xs:nonNegativeInteger | ipc_recv_volume - Interprocess communication recv volume |
| ipc_recv_wait_time | xs:nonNegativeInteger | ipc_recv_wait_time - Interprocess communication recv wait time |
| ipc_recvs_total | xs:nonNegativeInteger | ipc_recvs_total - Interprocess communication recvs total |
| ipc_send_volume | xs:nonNegativeInteger | ipc_send_volume - Interprocess communication send volume |

Table 123. Detailed metrics returned for MON_GET_CONNECTION_DETAILS (continued)

| Element Name | Data Type | Description |
|---------------------------------------|-----------------------|---|
| ipc_send_wait_time | xs:nonNegativeInteger | ipc_send_wait_time - Interprocess communication send wait time |
| ipc_sends_total | xs:nonNegativeInteger | ipc_sends_total - Interprocess communication send total |
| is_system_appl | xs:short | is_system_appl - Is System Application monitor element |
| last_executable_id | xs:hexBinary(32) | last_executable_id - Last executable identifier |
| last_request_type | xs:string(32) | last_request_type - Last request type |
| lob_prefetch_reqs | xs:nonNegativeInteger | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| lob_prefetch_wait_time | xs:nonNegativeInteger | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| lock_escals | xs:nonNegativeInteger | lock_escals - Number of lock escalations |
| lock_timeout_val | xs:long | lock_timeout_val - Lock timeout value monitor element |
| lock_timeouts | xs:nonNegativeInteger | lock_timeouts - Number of lock timeouts |
| lock_wait_time | xs:nonNegativeInteger | lock_wait_time - Time waited on locks |
| lock_waits | xs:nonNegativeInteger | lock_waits - Lock waits |
| log_buffer_wait_time | xs:nonNegativeInteger | log_buffer_wait_time - Log buffer wait time |
| log_disk_wait_time | xs:nonNegativeInteger | log_disk_wait_time - Log disk wait time |
| log_disk_waits_total | xs:nonNegativeInteger | log_disk_waits_total - Log disk waits total |
| member | xs:nonNegativeInteger | member - Database member |
| member_subset_id | xs:long | member_subset_id - Member subset ID |
| merge_sql_stmts | xs:long | merge_sql_stmts - Merge SQL statements executed |
| num_assoc_agents | xs:long | num_assoc_agents - Number of Associated Agents monitor element |
| num_locks_held | xs:nonNegativeInteger | locks_held - Locks held |
| num_locks_waiting | xs:long | locks_waiting - Current agents waiting on locks monitor element |
| num_log_buffer_full | xs:nonNegativeInteger | num_log_buffer_full - Number of full log buffers |
| num_lw_thresh_exceeded | xs:nonNegativeInteger | num_lw_thresh_exceeded - Number of thresholds exceeded |
| olap_func_overflows | xs:long | olap_func_overflows - OLAP Function Overflows monitor element |
| pkg_cache_inserts | xs:nonNegativeInteger | pkg_cache_inserts - Package cache inserts |
| pkg_cache_lookups | xs:nonNegativeInteger | pkg_cache_lookups - Package cache lookups |
| pool_col_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |
| pool_col_gbp_invalid_pages | xs:nonNegativeInteger | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| pool_col_gbp_l_reads | xs:nonNegativeInteger | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| pool_col_gbp_p_reads | xs:nonNegativeInteger | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| pool_col_l_reads | xs:nonNegativeInteger | pool_col_l_reads - Buffer pool column-organized logical reads |
| pool_col_lbp_pages_found | xs:nonNegativeInteger | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found |
| pool_col_p_reads | xs:nonNegativeInteger | pool_col_p_reads - Buffer pool column-organized physical reads |
| pool_col_writes | xs:nonNegativeInteger | pool_col_writes - Buffer pool column-organized writes |

Table 123. Detailed metrics returned for MON_GET_CONNECTION_DETAILS (continued)

| Element Name | Data Type | Description |
|---|-----------------------|--|
| pool_data_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| pool_data_l_reads | xs:nonNegativeInteger | pool_data_l_reads - Buffer pool data logical reads |
| pool_data_p_reads | xs:nonNegativeInteger | pool_data_p_reads - Buffer pool data physical reads |
| pool_data_writes | xs:nonNegativeInteger | pool_data_writes - Buffer pool data writes |
| pool_failed_async_col_reqs | xs:nonNegativeInteger | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| pool_failed_async_data_reqs | xs:nonNegativeInteger | pool_failed_async_data_reqs - Failed data prefetch requests |
| pool_failed_async_index_reqs | xs:nonNegativeInteger | pool_failed_async_index_reqs - Failed index prefetch requests |
| pool_failed_async_temp_col_reqs | xs:nonNegativeInteger | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |
| pool_failed_async_temp_data_reqs | xs:nonNegativeInteger | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces |
| pool_failed_async_temp_index_reqs | xs:nonNegativeInteger | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces |
| pool_failed_async_temp_xda_reqs | xs:nonNegativeInteger | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces |
| pool_failed_async_other_reqs | xs:nonNegativeInteger | pool_failed_async_other_reqs - Failed non-prefetch requests |
| pool_failed_async_xda_reqs | xs:nonNegativeInteger | pool_failed_async_xda_reqs - Failed XDA prefetch requests |
| pool_index_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| pool_index_l_reads | xs:nonNegativeInteger | pool_index_l_reads - Buffer pool index logical reads |
| pool_index_p_reads | xs:nonNegativeInteger | pool_index_p_reads - Buffer pool index physical reads |
| pool_index_writes | xs:nonNegativeInteger | pool_index_writes - Buffer pool index writes |
| pool_queued_async_col_reqs | xs:nonNegativeInteger | pool_queued_async_col_reqs - Column-organized prefetch requests |
| pool_queued_async_data_pages | xs:nonNegativeInteger | pool_queued_async_data_pages - Data pages prefetch requests |
| pool_queued_async_data_reqs | xs:nonNegativeInteger | pool_queued_async_data_reqs - Data prefetch requests |
| pool_queued_async_index_reqs | xs:nonNegativeInteger | pool_queued_async_index_reqs - Index prefetch requests |
| pool_queued_async_index_pages | xs:nonNegativeInteger | pool_queued_async_index_pages - Index pages prefetch requests |
| pool_queued_async_col_pages | xs:nonNegativeInteger | pool_queued_async_col_pages - Column-organized page prefetch requests |
| pool_queued_async_temp_col_pages | xs:nonNegativeInteger | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| pool_queued_async_temp_col_reqs | xs:nonNegativeInteger | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| pool_queued_async_temp_data_pages | xs:nonNegativeInteger | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces |
| pool_queued_async_temp_data_reqs | xs:nonNegativeInteger | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces |
| pool_queued_async_temp_index_pages | xs:nonNegativeInteger | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces |
| pool_queued_async_temp_index_reqs | xs:nonNegativeInteger | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces |
| pool_queued_async_temp_xda_pages | xs:nonNegativeInteger | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces |

Table 123. Detailed metrics returned for MON_GET_CONNECTION_DETAILS (continued)

| Element Name | Data Type | Description |
|---------------------------------------|-----------------------|--|
| pool_queued_async_temp_xda_reqs | xs:nonNegativeInteger | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces |
| pool_queued_async_other_reqs | xs:nonNegativeInteger | pool_queued_async_other_reqs - Non-prefetch requests |
| pool_queued_async_xda_reqs | xs:nonNegativeInteger | pool_queued_async_xda_reqs - XDA prefetch requests |
| pool_read_time | xs:nonNegativeInteger | pool_read_time - Total buffer pool physical read time |
| pool_temp_col_l_reads | xs:nonNegativeInteger | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| pool_temp_col_p_reads | xs:nonNegativeInteger | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| pool_temp_data_l_reads | xs:nonNegativeInteger | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| pool_temp_data_p_reads | xs:nonNegativeInteger | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| pool_temp_index_l_reads | xs:nonNegativeInteger | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| pool_temp_index_p_reads | xs:nonNegativeInteger | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| pool_temp_xda_l_reads | xs:nonNegativeInteger | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |
| pool_temp_xda_p_reads | xs:nonNegativeInteger | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| pool_write_time | xs:nonNegativeInteger | pool_write_time - Total buffer pool physical write time |
| pool_xda_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |
| pool_xda_gbp_invalid_pages | xs:nonNegativeInteger | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| pool_xda_gbp_l_reads | xs:nonNegativeInteger | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |
| pool_xda_gbp_p_reads | xs:nonNegativeInteger | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |
| pool_xda_l_reads | xs:nonNegativeInteger | pool_xda_l_reads - Buffer pool XDA data logical reads |
| pool_xda_lbp_pages_found | xs:nonNegativeInteger | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |
| pool_xda_p_reads | xs:nonNegativeInteger | pool_xda_p_reads - Buffer pool XDA data physical reads |
| pool_xda_writes | xs:nonNegativeInteger | pool_xda_writes - Buffer pool XDA data writes |
| post_shrthreshold_hash_joins | xs:long | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| post_shrthreshold_sorts | xs:nonNegativeInteger | post_shrthreshold_sorts - Post shared threshold sorts |
| post_threshold_col_vector_consumers | xs:long | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| post_threshold_hash_grpbys | xs:long | post_threshold_hash_grpbys - Hash group by threshold |
| post_threshold_hash_joins | xs:long | post_threshold_hash_joins - Hash Join Threshold monitor element |
| post_threshold_olap_funcs | xs:long | post_threshold_olap_funcs - OLAP function threshold monitor element |
| post_threshold_peas | xs:long | post_threshold_peas - Partial early aggregation threshold |
| post_threshold_peds | xs:long | post_threshold_peds - Partial early distincts threshold |
| post_threshold_sorts | xs:nonNegativeInteger | post_threshold_sorts - Post threshold sorts |
| prefetch_wait_time | xs:nonNegativeInteger | prefetch_wait_time - Time waited for prefetch |
| prefetch_waits | xs:nonNegativeInteger | prefetch_waits - Prefetcher wait count |

Table 123. Detailed metrics returned for MON_GET_CONNECTION_DETAILS (continued)

| Element Name | Data Type | Description |
|--------------------------------|-----------------------|--|
| prev_uow_stop_time | xs:dateTime | prev_uow_stop_time - Previous Unit of Work Completion Timestamp monitor element |
| reusability_status_reason | xs:string (255) | reusability_status_reason - Reusability status reason monitor element |
| rows_deleted | xs:long | rows_deleted - Rows deleted monitor element |
| rows_inserted | xs:long | rows_inserted - Rows inserted monitor element |
| rows_modified | xs:nonNegativeInteger | rows_modified - Rows modified |
| rows_read | xs:nonNegativeInteger | rows_read - Rows read |
| rows_returned | xs:nonNegativeInteger | rows_returned - Rows returned |
| rows_updated | xs:long | rows_updated - Rows updated monitor element |
| rqsts_completed_total | xs:nonNegativeInteger | rqsts_completed_total - Total requests completed |
| select_sql_stmts | xs:long | select_sql_stmts - Select SQL Statements Executed monitor element |
| session_auth_id | xs:string | session_auth_id - Session authorization ID |
| sort_heap_allocated | xs:long | sort_heap_allocated - Total Sort Heap Allocated |
| sort_overflows | xs:nonNegativeInteger | sort_overflows - Sort overflows |
| sort_shrheap_allocated | xs:long | sort_shrheap_allocated - Sort Share Heap Currently Allocated |
| spacemappage_reclaim_wait_time | xs:long | spacemappage_reclaim_wait_time - Space map page reclaim wait time |
| spacemappage_reclaim_wait_time | xs:long | spacemappage_reclaim_wait_time - Space map page reclaim wait time |
| sql_reqs_since_commit | xs:long | sql_reqs_since_commit - SQL Requests Since Last Commit monitor element |
| static_sql_stmts | xs:long | static_sql_stmts - Static SQL Statements Attempted monitor element |
| system_auth_id | xs:string | system_auth_id - System authorization identifier |
| tcpip_rcv_volume | xs:nonNegativeInteger | tcpip_rcv_volume - TCP/IP received volume |
| tcpip_rcv_wait_time | xs:nonNegativeInteger | tcpip_rcv_wait_time - TCP/IP rcv wait time |
| tcpip_rcvs_total | xs:nonNegativeInteger | tcpip_rcvs_total - TCP/IP rcvs total |
| tcpip_send_volume | xs:nonNegativeInteger | tcpip_send_volume - TCP/IP send volume |
| tcpip_send_wait_time | xs:nonNegativeInteger | tcpip_send_wait_time - TCP/IP send wait time |
| tcpip_sends_total | xs:nonNegativeInteger | tcpip_sends_total - TCP/IP sends total |
| thresh_violations | xs:nonNegativeInteger | thresh_violations - Number of threshold violations |
| total_act_time | xs:nonNegativeInteger | total_act_time - Total activity time |
| total_act_wait_time | xs:nonNegativeInteger | total_act_wait_time - Total activity wait time |
| total_app_commits | xs:nonNegativeInteger | total_app_commits - Total application commits |
| total_app_rollbacks | xs:nonNegativeInteger | total_app_rollbacks - Total application rollbacks |
| total_app_rqst_time | xs:nonNegativeInteger | total_app_rqst_time - Total application request time |
| total_app_section_executions | xs:nonNegativeInteger | total_app_section_executions - Total section executions |
| total_backup_proc_time | xs:long | total_backup_proc_time - Total non-wait time for online backups |
| total_backup_time | xs:long | total_backup_time - Total elapsed time for online backups |
| total_backups | xs:long | total_backups - Total backups |
| total_col_executions | xs:nonNegativeInteger | total_col_executions - Total column-organized executions |
| total_col_proc_time | xs:nonNegativeInteger | total_col_proc_time - Total column-organized processing time |
| total_col_synopsis_executions | xs:nonNegativeInteger | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |

Table 123. Detailed metrics returned for MON_GET_CONNECTION_DETAILS (continued)

| Element Name | Data Type | Description |
|--|-----------------------|--|
| total_col_synopsis_proc_time | xs:nonNegativeInteger | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| total_col_synopsis_time | xs:nonNegativeInteger | total_col_synopsis_time - Total column-organized synopsis table time monitor element |
| total_col_time | xs:nonNegativeInteger | total_col_time - Total column-organized time |
| total_col_vector_consumers | xs:long | total_col_vector_consumers - Total columnar_vector consumers |
| total_commit_proc_time | xs:nonNegativeInteger | total_commit_proc_time - Total commits processing time |
| total_commit_time | xs:nonNegativeInteger | total_commit_time - Total commit time |
| total_compilations | xs:nonNegativeInteger | total_compilations - Total compilations |
| total_compile_proc_time | xs:nonNegativeInteger | total_compile_proc_time - Total compile processing time |
| total_compile_time | xs:nonNegativeInteger | total_compile_time - Total compile time |
| total_connect_authentication_time | xs:nonNegativeInteger | total_connect_authentication_time - Total connection or switch user authentication request time monitor element |
| total_connect_authentication_proc_time | xs:nonNegativeInteger | total_connect_authentication_proc_time - Total connection authentication processing time monitor element |
| total_connect_authentications | xs:nonNegativeInteger | total_connect_authentications - Connections or switch user authentications performed |
| total_connect_request_proc_time | xs:nonNegativeInteger | total_connect_request_proc_time - Total connection or switch user request processing time monitor element |
| total_connect_request_time | xs:nonNegativeInteger | total_connect_request_time - Total connection or switch user request time monitor element |
| total_connect_requests | xs:nonNegativeInteger | total_connect_requests - Connection or switch user requests |
| total_cpu_time | xs:nonNegativeInteger | total_cpu_time - Total CPU time |
| total_disp_run_queue_time | xs:long | total_disp_run_queue_time - Total dispatcher run queue time |
| total_extended_latch_wait_time | xs:nonNegativeInteger | total_extended_latch_wait_time - Total extended latch wait time |
| total_extended_latch_waits | xs:nonNegativeInteger | total_extended_latch_waits - Total extended latch waits |
| total_hash_grpbys | xs:long | total_hash_grpbys - Total hash group by operations |
| total_hash_joins | xs:long | total_hash_joins - Total Hash Joins monitor element |
| total_hash_loops | xs:long | total_hash_loops - Total Hash Loops monitor element |
| total_implicit_compilations | xs:nonNegativeInteger | total_implicit_compilations - Total implicit complications |
| total_implicit_compile_proc_time | xs:nonNegativeInteger | total_implicit_compile_proc_time - Total implicit compile processing time |
| total_implicit_compile_time | xs:nonNegativeInteger | total_implicit_compile_time - Total implicit compile time |
| total_index_build_proc_time | xs:long | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation, recreation or rebuilding |
| total_index_build_time | xs:long | total_index_build_time - Total time spent building indexes due to index creation, recreation or rebuilding |
| total_indexes_built | xs:long | total_indexes_built - Total number of indexes built |
| total_load_proc_time | xs:nonNegativeInteger | total_load_proc_time - Total load processing time |
| total_load_time | xs:nonNegativeInteger | total_load_time - Total load time |
| total_loads | xs:nonNegativeInteger | total_loads - Total loads |
| total_olap_funcs | xs:long | total_olap_funcs - Total OLAP Functions monitor element |
| total_peas | xs:long | total_peas - Total partial early aggregations |
| total_peds | xs:long | total_peds - Total partial early distincts |
| total_reorg_proc_time | xs:nonNegativeInteger | total_reorg_proc_time - Total reorganization processing time |
| total_reorg_time | xs:nonNegativeInteger | total_reorg_time - Total reorganization time |

Table 123. Detailed metrics returned for MON_GET_CONNECTION_DETAILS (continued)

| Element Name | Data Type | Description |
|-----------------------------------|-----------------------|--|
| total_reorgs | xs:nonNegativeInteger | total_reorgs - Total reorganizations |
| total_rollback_proc_time | xs:nonNegativeInteger | total_rollback_proc_time - Total rollback processing time |
| total_rollback_time | xs:nonNegativeInteger | total_rollback_time - Total rollback time |
| total_routine_invocations | xs:nonNegativeInteger | total_routine_invocations - Total routine invocations |
| total_routine_time | xs:nonNegativeInteger | total_routine_time - Total routine time |
| total_routine_user_code_proc_time | xs:nonNegativeInteger | total_routine_user_code_proc_time - Total routine user code processing time |
| total_routine_user_code_time | xs:nonNegativeInteger | total_routine_user_code_time - Total routine user code time |
| total_rqst_time | xs:nonNegativeInteger | total_rqst_time - Total request time |
| total_runstats | xs:nonNegativeInteger | total_runstats - Total runtime statistics |
| total_runstats_proc_time | xs:nonNegativeInteger | total_runstats_proc_time - Total runtime statistics processing time |
| total_runstats_time | xs:nonNegativeInteger | total_runstats_time - Total runtime statistics |
| total_section_proc_time | xs:nonNegativeInteger | total_section_proc_time - Total section processing time |
| total_section_sort_proc_time | xs:nonNegativeInteger | total_section_sort_proc_time - Total section sort processing time |
| total_section_sort_time | xs:nonNegativeInteger | total_section_sort_time - Total section sort time |
| total_section_sorts | xs:nonNegativeInteger | total_section_sorts - Total section sorts |
| total_section_time | xs:nonNegativeInteger | total_section_time - Total section time |
| total_sorts | xs:nonNegativeInteger | total_sorts - Total Sorts |
| total_stats_fabrication_proc_time | xs:nonNegativeInteger | total_stats_fabrication_proc_time - Total statistics fabrication processing time |
| total_stats_fabrication_time | xs:nonNegativeInteger | total_stats_fabrication_time - Total statistics fabrication time |
| total_stats_fabrications | xs:nonNegativeInteger | total_stats_fabrications - Total statistics fabrications |
| total_sync_runstats | xs:nonNegativeInteger | total_sync_runstats - Total synchronous RUNSTATS activities |
| total_sync_runstats_proc_time | xs:nonNegativeInteger | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time |
| total_sync_runstats_time | xs:nonNegativeInteger | total_sync_runstats_time - Total synchronous RUNSTATS time |
| total_wait_time | xs:nonNegativeInteger | total_wait_time - Total wait time |
| tq_sort_heap_rejections | xs:long | tq_sort_heap_rejections - Table queue sort heap rejections |
| tq_sort_heap_requests | xs:long | tq_sort_heap_requests - Table queue sort heap requests |
| tq_tot_send_spills | xs:nonNegativeInteger | tq_tot_send_spills - Total number of table queue buffers overflowed |
| uid_sql_stmts | xs:long | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| uow_client_idle_wait_time | xs:long | uow_client_idle_wait_time - Client idle time within a unit of work |
| uow_comp_status | xs:string | uow_comp_status - Unit of Work Completion Status monitor element |
| uow_start_time | xs:dateTime | uow_start_time - Unit of work start timestamp monitor element |
| uow_stop_time | xs:dateTime | uow_stop_time - Unit of work stop timestamp monitor element |
| wlm_queue_assignments_total | xs:nonNegativeInteger | wlm_queue_assignments_total - Workload manager total queue assignments |
| wlm_queue_time_total | xs:nonNegativeInteger | wlm_queue_time_total - Workload manager total queue time |
| workload_occurrence_state | xs:string | workload_occurrence_state - Workload occurrence state monitor element |

Table 123. Detailed metrics returned for MON_GET_CONNECTION_DETAILS (continued)

| Element Name | Data Type | Description |
|--------------|-----------|--|
| xquery_stmts | xs:long | xquery_stmts - XQuery Statements Attempted monitor element |

Usage notes

The metrics returned by the MON_GET_CONNECTION_DETAILS table function represent the accumulation of all metrics for requests that were submitted by a connection. This function is similar to the MON_GET_CONNECTION table function:

- The MON_GET_CONNECTION table function returns the most commonly used metrics in a column-based format and is the most performance efficient method of retrieving metrics.
- The MON_GET_CONNECTION_DETAILS table function returns the entire set of available metrics in an XML document format, which provides maximum flexibility for formatting output. The XML-based output can be parsed directly by an XML parser, or it can be converted to relational format by the XMLTABLE function (see the example).

Metrics are rolled up at unit of work boundaries, and periodically during the execution of requests. Therefore, the values reported by this table function reflect the current state of the system at the time of the most recent rollup. Metrics are strictly increasing in value. To determine the value of a given metric for an interval of time, use the MON_GET_CONNECTION_DETAILS table function to query the metric at the start and end of the interval, and compute the difference.

Request metrics are controlled through the COLLECT REQUEST METRICS clause on service superclasses and the **mon_req_metrics** database configuration parameter at the database level. Metrics are only collected for a request if the request is processed by an agent in a service subclass whose parent service superclass has request metrics enabled, or if request metrics collection is enabled for the entire database. By default, request metrics are enabled at the database level. If request metrics are disabled at the database level, and for a service superclass, the metrics reported for each connection mapped to that service superclass stop increasing (or remain at 0 if request metrics were disabled at database activation time).

Tip: As a connection can be mapped to more than one service superclass during its lifetime, if collection is disabled at the database level, the metrics reported through the MON_GET_CONNECTION_DETAILS table function might represent a subset of the metrics for all requests submitted over the connection. This might occur if metrics collection is disabled for some of the superclasses to which the connection maps.

The MON_GET_CONNECTION_DETAILS table function returns one row of data per connection and per member. No aggregation across members (for a service class or more) is performed. However, aggregation can be achieved through SQL queries.

The schema for the XML document that is returned in the DETAILS column is available in the file `sql1lib/misc/DB2MonRoutines.xsd`. Further details can be found in the file `sql1lib/misc/DB2MonCommon.xsd`.

Example

Display connections returning the highest volume of data to clients, ordered by rows returned.

```
SELECT detmetrics.application_handle,
       detmetrics.rows_returned,
       detmetrics.tcpip_send_volume
FROM TABLE(MON_GET_CONNECTION_DETAILS(CAST(NULL as bigint), -2))
AS CONNMETRICS,
XMLTABLE (XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'),
          '$detmetric/db2_connection' PASSING XMLPARSE(DOCUMENT CONNMETRICS.DETAILS)
          as "detmetric"
COLUMNS "APPLICATION_HANDLE" INTEGER PATH 'application_handle',
         "ROWS_RETURNED" BIGINT PATH 'system_metrics/rows_returned',
         "TCP_IP_SEND_VOLUME" BIGINT PATH 'system_metrics/tcpip_send_volume'
        ) AS DETMETRICS
ORDER BY rows_returned DESC
```

The following is an example of output from this query.

| APPLICATION_HANDLE | ROWS_RETURNED | TCP_IP_SEND_VOLUME |
|--------------------|---------------|--------------------|
| 21 | 4 | 0 |

1 record(s) selected.

MON_GET_CONTAINER table function - Get table space container metrics

The MON_GET_CONTAINER table function returns monitor metrics for one or more table space containers.

Syntax

►►—MON_GET_CONTAINER—(—*tbsp_name*—,—*member*—)—►►

The schema is SYSPROC.

Table function parameters

tbsp_name

An input argument of type VARCHAR(128) that specifies a valid table space name in the same database as the one currently connected to when calling this function. If the argument is null or an empty string, metrics are returned for all containers in all table spaces in the database.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

Example 1: List containers on all database members that have the highest read time.

```
SELECT varchar(container_name,70) as container_name,  
       varchar(tbsp_name,20) as tbsp_name,  
       pool_read_time  
FROM TABLE(MON_GET_CONTAINER('','-2')) AS t  
ORDER BY pool_read_time DESC
```

The following is an example of output from this query.

```
CONTAINER_NAME  
-----  
/home/hotel55/swalkty/swalkty/NODE0000/TEST/T0000000/C0000000.CAT ...  
/home/hotel55/swalkty/swalkty/NODE0000/TEST/T0000002/C0000000.LRG ...  
/home/hotel55/swalkty/swalkty/NODE0000/TEST/T0000001/C0000000.TMP ...
```

3 record(s) selected.

Output for query (continued).

```
... TBSP_NAME          POOL_READ_TIME  
... -----  
... SYSCATSPACE                597  
... USERSPACE1                 42  
... TEMPSPACE1                 0
```

Example 2: List any containers that are not accessible.

```
SELECT varchar(container_name, 70) as container_name  
FROM TABLE(MON_GET_CONTAINER('','-1)) AS t  
WHERE accessible = 0
```

The following is an example of output from this query.

```
CONTAINER_NAME  
-----
```

0 record(s) selected.

Example 3: List utilization of container file systems, ordered by highest utilization.

```
SELECT varchar(container_name, 65) as container_name,  
       fs_id,  
       fs_used_size,  
       fs_total_size,  
       CASE WHEN fs_total_size > 0  
            THEN DEC(100*(FLOAT(fs_used_size)/FLOAT(fs_total_size)),5,2)  
            ELSE DEC(-1,5,2)  
       END as utilization  
FROM TABLE(MON_GET_CONTAINER('','-1)) AS t  
ORDER BY utilization DESC
```

The following is an example of output from this query.

```
CONTAINER_NAME  
-----  
/home/hotel55/swalkty/swalkty/NODE0000/TEST/T0000000/C0000000.CAT ...  
/home/hotel55/swalkty/swalkty/NODE0000/TEST/T0000001/C0000000.TMP ...  
/home/hotel55/swalkty/swalkty/NODE0000/TEST/T0000002/C0000000.LRG ...
```

3 record(s) selected.

Output for query (continued).

| FS_ID | FS_USED_SIZE | FS_TOTAL_SIZE | UTILIZATION |
|-------|--------------|---------------|-------------|
| 64768 | 106879311872 | 317068410880 | 33.70 |
| 64768 | 106879311872 | 317068410880 | 33.70 |
| 64768 | 106879311872 | 317068410880 | 33.70 |

Usage notes

The MON_GET_CONTAINER table function returns one row of data per container and per database member. Data can be returned for all containers in a given table space, or for all containers in the database. No aggregation across database partitions is performed. However, aggregation can be achieved through SQL queries.

Metrics collected by this function are controlled at the database level using the mon_obj_metrics configuration parameter. By default, metrics collection is enabled.

Information returned

Table 124. Information returned for MON_GET_CONTAINER

| Column Name | Data Type | Description or corresponding monitor element |
|-------------------------|--------------|---|
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container name |
| CONTAINER_ID | BIGINT | container_id - Container identification |
| MEMBER | SMALLINT | member - Database member |
| CONTAINER_TYPE | VARCHAR(16) | container_type - Container type This is a text identifier based on the defines in sqlutil.h and is one of: <ul style="list-style-type: none"> • DISK_EXTENT_TAG • DISK_PAGE_TAG • FILE_EXTENT_TAG • FILE_PAGE_TAG • PATH |
| STRIPE_SET | BIGINT | container_stripe_set - Stripe set |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| PAGES_READ | BIGINT | pages_read - Number of pages read |
| PAGES_WRITTEN | BIGINT | pages_written - Number of pages written |
| VECTORED_IOS | BIGINT | vectored_ios - Number of vectored IO requests |
| PAGES_FROM_VECTORED_IOS | BIGINT | pages_from_vectored_ios - Total number of pages read by vectored IO |
| BLOCK_IOS | BIGINT | block_ios - Number of block IO requests |
| PAGES_FROM_BLOCK_IOS | BIGINT | pages_from_block_ios - Total number of pages read by block IO |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |

Table 124. Information returned for MON_GET_CONTAINER (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|--------------------|-------------|--|
| TOTAL_PAGES | BIGINT | container_total_pages - Total pages in container |
| USABLE_PAGES | BIGINT | container_usable_pages - Usable pages in container |
| ACCESSIBLE | SMALLINT | container_accessible - Accessibility of container |
| FS_ID | VARCHAR(22) | fs_id - Unique file system identification number |
| FS_TOTAL_SIZE | BIGINT | fs_total_size - Total size of a file system |
| FS_USED_SIZE | BIGINT | fs_used_size - Amount of space used on a file system |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element

For a partitioned database environment, this will be the same value as for the MEMBER column. For Db2 Enterprise Server Edition and in a Db2 pureScale environment, this value will be 0.
Note: DBPARTITIONNUM is different to data_partition_id , which is used to identify a data partition that was created by subdividing data in a table based on a value. |
| DB_STORAGE_PATH_ID | BIGINT | db_storage_path_id - Storage path identifier monitor element |

MON_GET_DATABASE table function - Get database metrics

The MON_GET_DATABASE table function returns database level information within the monitor infrastructure.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_GET_DATABASE—(—member—)—————►

The schema is SYSPROC.

Routine parameters

member

An input parameter of type INTEGER that specifies a valid database member from which the records should be fetched. Specify -1 or null for the current member, or -2 for information from all active database members. An active database member is a database that is available for connection and use by applications.

Information returned

Table 125. Information returned by the MON_GET_DATABASE routine

| Column name | Data type | Description |
|-------------------------|---------------|--|
| MEMBER | SMALLINT | member - Database member monitor element |
| DB_STATUS | VARCHAR(16) | db_status - Status of database monitor element |
| DB_ACTIVATION_STATE | VARCHAR(10) | db_activation_state - Database activation state |
| DB_CONN_TIME | TIMESTAMP | db_conn_time - Database activation timestamp monitor element |
| CATALOG_PARTITION | SMALLINT | catalog_node - Catalog Node Number |
| LAST_BACKUP | TIMESTAMP | last_backup - Last Backup Timestamp monitor element |
| CONNECTIONS_TOP | BIGINT | connections_top - Maximum Number of Concurrent Connections monitor element |
| TOTAL_CONS | BIGINT | total_cons - Connects Since Database Activation monitor element |
| TOTAL_SEC_CONS | BIGINT | total_sec_cons - Secondary Connections monitor element |
| APPLS_CUR_CONS | BIGINT | appls_cur_cons - Applications Connected Currently monitor element |
| APPLS_IN_DB2 | BIGINT | appls_in_db2 - Applications Executing in the Database Currently monitor element |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of Associated Agents monitor element |
| AGENTS_TOP | BIGINT | agents_top - Number of Agents Created monitor element |
| NUM_COORD_AGENTS | BIGINT | num_coord_agents - Number of coordinator agents |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum Number of Coordinating Agents monitor element |
| NUM_LOCKS_HELD | BIGINT | locks_held - Locks held monitor element |
| NUM_LOCKS_WAITING | BIGINT | locks_waiting - Current agents waiting on locks monitor element |
| LOCK_LIST_IN_USE | BIGINT | lock_list_in_use - Total lock list memory in use monitor element |
| ACTIVE_SORTS | BIGINT | active_sorts - Active Sorts monitor element |
| ACTIVE_HASH_JOINS | BIGINT | active_hash_joins - Active hash joins monitor element |
| ACTIVE_OLAP_FUNCS | BIGINT | active_olap_funcs - Active OLAP Functions monitor element |
| DB_PATH | VARCHAR(1024) | db_path - Database Path monitor element |
| ACT_ABORTED_TOTAL | BIGINT | act_aborted_total - Total aborted activities monitor element |
| ACT_COMPLETED_TOTAL | BIGINT | act_completed_total - Total completed activities monitor element |
| ACT_REJECTED_TOTAL | BIGINT | act_rejected_total - Total rejected activities monitor element |
| AGENT_WAIT_TIME | BIGINT | agent_wait_time - Agent wait time monitor element |
| AGENT_WAITS_TOTAL | BIGINT | agent_waits_total - Total agent waits monitor element |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads monitor element |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads monitor element |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads monitor element |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer pool XDA data logical reads monitor element |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|-------------------------|-----------|---|
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads monitor element |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads monitor element |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads monitor element |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads monitor element |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer pool XDA data physical reads monitor element |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes monitor element |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes monitor element |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer pool XDA data writes monitor element |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time monitor element |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time monitor element |
| CLIENT_IDLE_WAIT_TIME | BIGINT | client_idle_wait_time - Client idle wait time monitor element |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected monitor element |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database monitor element |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database monitor element |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time monitor element |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests monitor element |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests monitor element |
| FCM_RECV_VOLUME | BIGINT | fcm_rcv_volume - FCM received volume monitor element |
| FCM_RECVS_TOTAL | BIGINT | fcm_rcvs_total - FCM receives total monitor element |
| FCM_SEND_VOLUME | BIGINT | fcm_send_volume - FCM send volume monitor element |
| FCM_SENDS_TOTAL | BIGINT | fcm_sends_total - FCM sends total monitor element |
| FCM_RECV_WAIT_TIME | BIGINT | fcm_rcv_wait_time - FCM received wait time monitor element |
| FCM_SEND_WAIT_TIME | BIGINT | fcm_send_wait_time - FCM send wait time monitor element |
| IPC_RECV_VOLUME | BIGINT | ipc_rcv_volume - Interprocess communication received volume monitor element |
| IPC_RECV_WAIT_TIME | BIGINT | ipc_rcv_wait_time - Interprocess communication received wait time monitor element |
| IPC_RECVS_TOTAL | BIGINT | ipc_rcvs_total - Interprocess communication receives total monitor element |
| IPC_SEND_VOLUME | BIGINT | ipc_send_volume - Interprocess communication send volume monitor element |
| IPC_SEND_WAIT_TIME | BIGINT | ipc_send_wait_time - Interprocess communication send wait time monitor element |
| IPC_SENDS_TOTAL | BIGINT | ipc_sends_total - Interprocess communication send total monitor element |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations monitor element |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|------------------------------|-----------|---|
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts monitor element |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks monitor element |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits monitor element |
| LOG_BUFFER_WAIT_TIME | BIGINT | log_buffer_wait_time - Log buffer wait time monitor element |
| NUM_LOG_BUFFER_FULL | BIGINT | num_log_buffer_full - Number of times full log buffer caused agents to wait monitor element |
| LOG_DISK_WAIT_TIME | BIGINT | log_disk_wait_time - Log disk wait time monitor element |
| LOG_DISK_WAITS_TOTAL | BIGINT | log_disk_waits_total - Total log disk waits monitor element |
| RQSTS_COMPLETED_TOTAL | BIGINT | rqsts_completed_total - Total requests completed monitor element |
| ROWS_MODIFIED | BIGINT | rows_modified - Rows modified monitor element |
| ROWS_READ | BIGINT | rows_read - Rows read monitor element |
| ROWS_RETURNED | BIGINT | rows_returned - Rows returned monitor element |
| TCPIP_RECV_VOLUME | BIGINT | tcPIP_recv_volume - TCP/IP received volume monitor element |
| TCPIP_SEND_VOLUME | BIGINT | tcPIP_send_volume - TCP/IP send volume monitor element |
| TCPIP_RECV_WAIT_TIME | BIGINT | tcPIP_recv_wait_time - TCP/IP received wait time monitor element |
| TCPIP_RECVS_TOTAL | BIGINT | tcPIP_recvS_total - TCP/IP receives total monitor element |
| TCPIP_SEND_WAIT_TIME | BIGINT | tcPIP_send_wait_time - TCP/IP send wait time monitor element |
| TCPIP_SENDS_TOTAL | BIGINT | tcPIP_sendS_total - TCP/IP sends total monitor element |
| TOTAL_APP_RQST_TIME | BIGINT | total_app_rqst_time - Total application request time monitor element |
| TOTAL_RQST_TIME | BIGINT | total_rqst_time - Total request time monitor element |
| WLM_QUEUE_TIME_TOTAL | BIGINT | wlm_queue_time_total - Workload manager total queue time monitor element |
| WLM_QUEUE_ASSIGNMENTS_TOTAL | BIGINT | wlm_queue_assignments_total - Workload manager total queue assignments monitor element |
| TOTAL_RQST_MAPPED_IN | BIGINT | total_rqst_mapped_in - Total request mapped-in monitor element |
| TOTAL_RQST_MAPPED_OUT | BIGINT | total_rqst_mapped_out - Total request mapped-out monitor element |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time monitor element |
| TOTAL_WAIT_TIME | BIGINT | total_wait_time - Total wait time monitor element |
| APP_RQSTS_COMPLETED_TOTAL | BIGINT | app_rqsts_completed_total - Total application requests completed monitor element |
| TOTAL_SECTION_SORT_TIME | BIGINT | total_section_sort_time - Total section sort time monitor element |
| TOTAL_SECTION_SORT_PROC_TIME | BIGINT | total_section_sort_proc_time - Total section sort processing time monitor element |
| TOTAL_SECTION_SORTS | BIGINT | total_section_sorts - Total section sorts monitor element |
| TOTAL_SORTS | BIGINT | total_sorts - Total sorts monitor element |
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post threshold sorts monitor element |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post shared threshold sorts monitor element |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows monitor element |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|----------------------------------|-----------|---|
| TOTAL_COMPILE_TIME | BIGINT | total_compile_time - Total compile time monitor element |
| TOTAL_COMPILE_PROC_TIME | BIGINT | total_compile_proc_time - Total compile processing time monitor element |
| TOTAL_COMPILATIONS | BIGINT | total_compilations - Total compilations monitor element |
| TOTAL_IMPLICIT_COMPILE_TIME | BIGINT | total_implicit_compile_time - Total implicit compile time monitor element |
| TOTAL_IMPLICIT_COMPILE_PROC_TIME | BIGINT | total_implicit_compile_proc_time - Total implicit compile processing time monitor element |
| TOTAL_IMPLICIT_COMPILATIONS | BIGINT | total_implicit_compilations - Total implicit complications monitor element |
| TOTAL_SECTION_TIME | BIGINT | total_section_time - Total section time monitor element |
| TOTAL_SECTION_PROC_TIME | BIGINT | total_section_proc_time - Total section processing time monitor element |
| TOTAL_APP_SECTION_EXECUTIONS | BIGINT | total_app_section_executions - Total application section executions monitor element |
| TOTAL_ACT_TIME | BIGINT | total_act_time - Total activity time monitor element |
| TOTAL_ACT_WAIT_TIME | BIGINT | total_act_wait_time - Total activity wait time monitor element |
| ACT_RQSTS_TOTAL | BIGINT | act_rqsts_total - Total activity requests monitor elements |
| TOTAL_ROUTINE_TIME | BIGINT | total_routine_time - Total routine time monitor element |
| TOTAL_ROUTINE_INVOCATIONS | BIGINT | total_routine_invocations - Total routine invocations monitor elements |
| TOTAL_COMMIT_TIME | BIGINT | total_commit_time - Total commit time monitor element |
| TOTAL_COMMIT_PROC_TIME | BIGINT | total_commit_proc_time - Total commits processing time monitor element |
| TOTAL_APP_COMMITS | BIGINT | total_app_commits - Total application commits monitor elements |
| INT_COMMITS | BIGINT | int_commits - Internal commits monitor element |
| TOTAL_ROLLBACK_TIME | BIGINT | total_rollback_time - Total rollback time monitor element |
| TOTAL_ROLLBACK_PROC_TIME | BIGINT | total_rollback_proc_time - Total rollback processing time monitor element |
| TOTAL_APP_ROLLBACKS | BIGINT | total_app_rollbacks - Total application rollbacks monitor element |
| INT_ROLLBACKS | BIGINT | int_rollbacks - Internal rollbacks monitor element |
| TOTAL_RUNSTATS_TIME | BIGINT | total_runstats_time - Total runtime statistics time monitor element |
| TOTAL_RUNSTATS_PROC_TIME | BIGINT | total_runstats_proc_time - Total runtime statistics processing time monitor element |
| TOTAL_RUNSTATS | BIGINT | total_runstats - Total runtime statistics monitor element |
| TOTAL_REORG_TIME | BIGINT | total_reorg_time - Total reorganization time monitor element |
| TOTAL_REORG_PROC_TIME | BIGINT | total_reorg_proc_time - Total reorganization processing time monitor element |
| TOTAL_REORGS | BIGINT | total_reorgs - Total reorganizations monitor element |
| TOTAL_LOAD_TIME | BIGINT | total_load_time - Total load time monitor element |
| TOTAL_LOAD_PROC_TIME | BIGINT | total_load_proc_time - Total load processing time monitor element |
| TOTAL_LOADS | BIGINT | total_loads - Total loads monitor element |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts monitor element |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|------------------------------------|-----------|--|
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups monitor element |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts monitor element |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups monitor element |
| THRESH_VIOLATIONS | BIGINT | thresh_violations - Number of threshold violations monitor element |
| NUM_LW_THRESH_EXCEEDED | BIGINT | num_lw_thresh_exceeded - Number of lock wait thresholds exceeded monitor element |
| LOCK_WAITS_GLOBAL | BIGINT | lock_waits_global - Lock waits global monitor element |
| LOCK_WAIT_TIME_GLOBAL | BIGINT | lock_wait_time_global - Lock wait time global monitor element |
| LOCK_TIMEOUTS_GLOBAL | BIGINT | lock_timeouts_global - Lock timeouts global monitor element |
| LOCK_ESCALS_MAXLOCKS | BIGINT | lock_escals_maxlocks - Number of maxlocks lock escalations monitor element |
| LOCK_ESCALS_LOCKLIST | BIGINT | lock_escals_locklist - Number of locklist lock escalations monitor element |
| LOCK_ESCALS_GLOBAL | BIGINT | lock_escals_global - Number of global lock escalations monitor element |
| DATA_SHARING_REMOTE_LOCKWAIT_COUNT | BIGINT | data_sharing_remote_lockwait_count - Data sharing remote lock wait count monitor element |
| DATA_SHARING_REMOTE_LOCKWAIT_TIME | BIGINT | data_sharing_remote_lockwait_time - Data sharing remote lock wait time monitor element |
| RECLAIM_WAIT_TIME | BIGINT | reclaim_wait_time - Reclaim wait time monitor element |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME | BIGINT | spacemappage_reclaim_wait_time - Space map page reclaim wait time monitor element |
| CF_WAITS | BIGINT | cf_waits - Number of cluster caching facility waits monitor element |
| CF_WAIT_TIME | BIGINT | cf_wait_time - cluster caching facility wait time monitor element |
| POOL_DATA_GBP_L_READS | BIGINT | pool_data_gbp_l_reads - Group buffer pool data logical reads monitor element |
| POOL_DATA_GBP_P_READS | BIGINT | pool_data_gbp_p_reads - Group buffer pool data physical reads monitor element |
| POOL_DATA_LBP_PAGES_FOUND | BIGINT | pool_data_lbp_pages_found - Local buffer pool found data pages monitor element |
| POOL_DATA_GBP_INVALID_PAGES | BIGINT | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages monitor element |
| POOL_INDEX_GBP_L_READS | BIGINT | pool_index_gbp_l_reads - Group buffer pool index logical reads monitor element |
| POOL_INDEX_GBP_P_READS | BIGINT | pool_index_gbp_p_reads - Group buffer pool index physical reads monitor elements |
| POOL_INDEX_LBP_PAGES_FOUND | BIGINT | pool_index_lbp_pages_found - Local buffer pool index pages found monitor element |
| POOL_INDEX_GBP_INVALID_PAGES | BIGINT | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages monitor element |
| POOL_XDA_GBP_L_READS | BIGINT | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests monitor element |
| POOL_XDA_GBP_P_READS | BIGINT | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests monitor element |
| POOL_XDA_LBP_PAGES_FOUND | BIGINT | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found monitor element |
| POOL_XDA_GBP_INVALID_PAGES | BIGINT | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages monitor element |
| AUDIT_EVENTS_TOTAL | BIGINT | audit_events_total - Total audit events monitor element |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|-----------------------------------|-----------|--|
| AUDIT_FILE_WRITES_TOTAL | BIGINT | audit_file_writes_total - Total audit files written monitor element |
| AUDIT_FILE_WRITE_WAIT_TIME | BIGINT | audit_file_write_wait_time - Audit file write wait time monitor element |
| AUDIT_SUBSYSTEM_WAITS_TOTAL | BIGINT | audit_subsystem_waits_total - Total audit subsystem waits monitor element |
| AUDIT_SUBSYSTEM_WAIT_TIME | BIGINT | audit_subsystem_wait_time - Audit subsystem wait time monitor element |
| DIAGLOG_WRITES_TOTAL | BIGINT | diaglog_writes_total - Total diagnostic log file writes monitor element |
| DIAGLOG_WRITE_WAIT_TIME | BIGINT | diaglog_write_wait_time - Diagnostic log file write wait time monitor element |
| FCM_MESSAGE_RECVS_TOTAL | BIGINT | fcm_message_recvs_total - Total FCM message receives monitor element |
| FCM_MESSAGE_RECV_VOLUME | BIGINT | fcm_message_recv_volume - FCM message received volume monitor element |
| FCM_MESSAGE_RECV_WAIT_TIME | BIGINT | fcm_message_recv_wait_time - FCM message received wait time monitor element |
| FCM_MESSAGE_SENDS_TOTAL | BIGINT | fcm_message_sends_total - Total FCM message sends monitor element |
| FCM_MESSAGE_SEND_VOLUME | BIGINT | fcm_message_send_volume - FCM message send volume monitor element |
| FCM_MESSAGE_SEND_WAIT_TIME | BIGINT | fcm_message_send_wait_time - FCM message send wait time monitor element |
| FCM_TQ_RECVS_TOTAL | BIGINT | fcm_tq_recvs_total - FCM table queue receives total monitor element |
| FCM_TQ_RECV_VOLUME | BIGINT | fcm_tq_recv_volume - FCM table queue received volume monitor element |
| FCM_TQ_RECV_WAIT_TIME | BIGINT | fcm_tq_recv_wait_time - FCM table queue received wait time monitor element |
| FCM_TQ_SENDS_TOTAL | BIGINT | fcm_tq_sends_total - FCM table queue send total monitor element |
| FCM_TQ_SEND_VOLUME | BIGINT | fcm_tq_send_volume - FCM table queue send volume monitor element |
| FCM_TQ_SEND_WAIT_TIME | BIGINT | fcm_tq_send_wait_time - FCM table queue send wait time monitor element |
| TOTAL_ROUTINE_USER_CODE_PROC_TIME | BIGINT | total_routine_user_code_proc_time - Total routine user code processing time monitor element |
| TOTAL_ROUTINE_USER_CODE_TIME | BIGINT | total_routine_user_code_time - Total routine user code time monitor element |
| TQ_TOT_SEND_SPILLS | BIGINT | tq_tot_send_spills - Total number of table queue buffers overflowed monitor element |
| EVMON_WAIT_TIME | BIGINT | evmon_wait_time - Event monitor wait time monitor element |
| EVMON_WAITS_TOTAL | BIGINT | evmon_waits_total - Event monitor total waits monitor element |
| TOTAL_EXTENDED_LATCH_WAIT_TIME | BIGINT | total_extended_latch_wait_time - Total extended latch wait time monitor element |
| TOTAL_EXTENDED_LATCH_WAITS | BIGINT | total_extended_latch_waits - Total extended latch waits monitor element |
| TOTAL_STATS_FABRICATION_TIME | BIGINT | total_stats_fabrication_time - Total statistics fabrication time monitor element |
| TOTAL_STATS_FABRICATION_PROC_TIME | BIGINT | total_stats_fabrication_proc_time - Total statistics fabrication processing time monitor element |
| TOTAL_STATS_FABRICATIONS | BIGINT | total_stats_fabrications - Total statistics fabrications monitor elements |
| TOTAL_SYNC_RUNSTATS_TIME | BIGINT | total_sync_runstats_time - Total synchronous RUNSTATS time monitor elements |
| TOTAL_SYNC_RUNSTATS_PROC_TIME | BIGINT | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time monitor element |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|------------------------------------|-----------|--|
| TOTAL_SYNC_RUNSTATS | BIGINT | total_sync_runstats - Total synchronous RUNSTATS activities monitor element |
| TOTAL_DISP_RUN_QUEUE_TIME | BIGINT | total_disp_run_queue_time - Total dispatcher run queue time monitor element |
| TOTAL_PEDS | BIGINT | total_peds - Total partial early distincts monitor element |
| DISABLED_PEDS | BIGINT | disabled_peds - Disabled partial early distincts monitor element |
| POST_THRESHOLD_PEDS | BIGINT | post_threshold_peds - Partial early distincts threshold monitor element |
| TOTAL_PEAS | BIGINT | total_peas - Total partial early aggregations monitor element |
| POST_THRESHOLD_PEAS | BIGINT | post_threshold_peas - Partial early aggregation threshold monitor element |
| TQ_SORT_HEAP_REQUESTS | BIGINT | tq_sort_heap_requests - Table queue sort heap requests monitor element |
| TQ_SORT_HEAP_REJECTIONS | BIGINT | tq_sort_heap_rejections - Table queue sort heap rejections monitor element |
| POOL_QUEUED_ASYNC_DATA_REQS | BIGINT | pool_queued_async_data_reqs - Data prefetch requests monitor element |
| POOL_QUEUED_ASYNC_INDEX_REQS | BIGINT | pool_queued_async_index_reqs - Index prefetch requests monitor element |
| POOL_QUEUED_ASYNC_XDA_REQS | BIGINT | pool_queued_async_xda_reqs - XDA prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_OTHER_REQS | BIGINT | pool_queued_async_other_reqs - Other requests handled by prefetchers monitor element |
| POOL_QUEUED_ASYNC_DATA_PAGES | BIGINT | pool_queued_async_data_pages - Data pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_INDEX_PAGES | BIGINT | pool_queued_async_index_pages - Index pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_XDA_PAGES | BIGINT | pool_queued_async_xda_pages - XDA pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES | BIGINT | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES | BIGINT | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_DATA_REQS | BIGINT | pool_failed_async_data_reqs - Failed data prefetch requests monitor element |
| POOL_FAILED_ASYNC_INDEX_REQS | BIGINT | pool_failed_async_index_reqs - Failed index prefetch requests monitor element |
| POOL_FAILED_ASYNC_XDA_REQS | BIGINT | pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element |
| POOL_FAILED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|---|-----------|--|
| POOL_FAILED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_OTHER_REQS | BIGINT | pool_failed_async_other_reqs - Failed non-prefetch requests monitor element |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch monitor element |
| PREFETCH_WAITS | BIGINT | prefetch_waits - Prefetcher wait count monitor element |
| APP_ACT_COMPLETED_TOTAL | BIGINT | app_act_completed_total - Total successful external coordinator activities monitor element |
| APP_ACT_ABORTED_TOTAL | BIGINT | app_act_aborted_total - Total failed external coordinator activities monitor element |
| APP_ACT_REJECTED_TOTAL | BIGINT | app_act_rejected_total - Total rejected external coordinator activities monitor element |
| TOTAL_CONNECT_REQUEST_TIME | BIGINT | total_connect_request_time - Total connection or switch user request time monitor element |
| TOTAL_CONNECT_REQUEST_PROC_TIME | BIGINT | total_connect_request_proc_time - Total connection or switch user request processing time monitor element |
| TOTAL_CONNECT_REQUESTS | BIGINT | total_connect_requests - Connection or switch user requests monitor element |
| TOTAL_CONNECT_AUTHENTICATION_TIME | BIGINT | total_connect_authentication_time - Total connection or switch user authentication request time monitor element |
| TOTAL_CONNECT_AUTHENTICATION_PROC_TIME | BIGINT | total_connect_authentication_proc_time - Total connection authentication processing time monitor element |
| TOTAL_CONNECT_AUTHENTIFICATIONS | BIGINT | total_connect_authentications - Connections or switch user authentications performed monitor element |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |
| COMM_EXIT_WAIT_TIME | BIGINT | comm_exit_wait_time - Communication buffer exit wait time monitor element |
| COMM_EXIT_WAITS | BIGINT | comm_exit_waits - Communication buffer exit number of waits monitor element |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads monitor element |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests monitor element |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes monitor element |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads monitor element |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests monitor element |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes monitor element |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer pool asynchronous XDA data reads monitor element |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer pool asynchronous XDA read requests monitor element |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer pool asynchronous XDA data writes monitor element |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|------------------------------------|-----------|---|
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer pool no victim buffers monitor element |
| POOL_LSN_GAP_CLNS | BIGINT | pool_lsn_gap_clns - Buffer pool log space cleaners triggered monitor element |
| POOL_DRTY_PG_STEAL_CLNS | BIGINT | pool_drty_pg_steal_clns - Buffer pool victim page cleaners triggered monitor element |
| POOL_DRTY_PG_THRSH_CLNS | BIGINT | pool_drty_pg_thrsh_clns - Buffer pool threshold cleaners triggered monitor element |
| VECTORED_IOS | BIGINT | vectored_ios - Number of vectored I/O requests monitor element |
| PAGES_FROM_VECTORED_IOS | BIGINT | pages_from_vectored_ios - Total number of pages read by vectored I/O monitor element |
| BLOCK_IOS | BIGINT | block_ios - Number of block I/O requests monitor element |
| PAGES_FROM_BLOCK_IOS | BIGINT | pages_from_block_ios - Total number of pages read by block I/O monitor element |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages monitor element |
| FILES_CLOSED | BIGINT | files_closed - Database files closed monitor element |
| POOL_ASYNC_DATA_GBP_L_READS | BIGINT | pool_async_data_gbp_l_reads - Asynchronous group buffer pool data logical reads monitor element |
| POOL_ASYNC_DATA_GBP_P_READS | BIGINT | pool_async_data_gbp_p_reads - Asynchronous group buffer pool data physical reads monitor element |
| POOL_ASYNC_DATA_LBP_PAGES_FOUND | BIGINT | pool_async_data_lbp_pages_found - Asynchronous local buffer pool data pages found monitor element |
| POOL_ASYNC_DATA_GBP_INVALID_PAGES | BIGINT | pool_async_data_gbp_invalid_pages - Asynchronous group buffer pool invalid data pages monitor element |
| POOL_ASYNC_INDEX_GBP_L_READS | BIGINT | pool_async_index_gbp_l_reads - Asynchronous group buffer pool index logical reads monitor element |
| POOL_ASYNC_INDEX_GBP_P_READS | BIGINT | pool_async_index_gbp_p_reads - Asynchronous group buffer pool index physical reads monitor element |
| POOL_ASYNC_INDEX_LBP_PAGES_FOUND | BIGINT | pool_async_index_lbp_pages_found - Asynchronous local buffer pool index pages found monitor element |
| POOL_ASYNC_INDEX_GBP_INVALID_PAGES | BIGINT | pool_async_index_gbp_invalid_pages - Asynchronous group buffer pool invalid index pages monitor element |
| POOL_ASYNC_XDA_GBP_L_READS | BIGINT | pool_async_xda_gbp_l_reads - Group buffer pool XDA data asynchronous logical read requests monitor element |
| POOL_ASYNC_XDA_GBP_P_READS | BIGINT | pool_async_xda_gbp_p_reads - Group buffer pool XDA data asynchronous physical read requests monitor element |
| POOL_ASYNC_XDA_LBP_PAGES_FOUND | BIGINT | pool_async_xda_lbp_pages_found - Asynchronous local buffer pool XDA data pages found monitor element |
| POOL_ASYNC_XDA_GBP_INVALID_PAGES | BIGINT | pool_async_xda_gbp_invalid_pages - Asynchronous group buffer pool invalid XDA data pages monitor element |
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer pool asynchronous read time |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time |
| SKIPPED_PREFETCH_DATA_P_READS | BIGINT | skipped_prefetch_data_p_reads - Skipped prefetch data physical reads monitor element |
| SKIPPED_PREFETCH_INDEX_P_READS | BIGINT | skipped_prefetch_index_p_reads - Skipped prefetch index physical reads monitor element |
| SKIPPED_PREFETCH_XDA_P_READS | BIGINT | skipped_prefetch_xda_p_reads - Skipped prefetch XDA physical reads monitor element |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|---|------------|---|
| SKIPPED_PREFETCH_TEMP_DATA_P_READS | BIGINT | skipped_prefetch_temp_data_p_reads - Skipped prefetch temporary data physical reads monitor element |
| SKIPPED_PREFETCH_TEMP_INDEX_P_READS | BIGINT | skipped_prefetch_temp_index_p_reads - Skipped prefetch temporary index physical reads monitor element |
| SKIPPED_PREFETCH_TEMP_XDA_P_READS | BIGINT | skipped_prefetch_temp_xda_p_reads - Skipped prefetch temporary XDA data physical reads monitor element |
| SKIPPED_PREFETCH_UOW_DATA_P_READS | BIGINT | skipped_prefetch_uow_data_p_reads - Skipped prefetch unit of work data physical reads monitor element |
| SKIPPED_PREFETCH_UOW_INDEX_P_READS | BIGINT | skipped_prefetch_uow_index_p_reads - Skipped prefetch unit of work index physical reads monitor element |
| SKIPPED_PREFETCH_UOW_XDA_P_READS | BIGINT | skipped_prefetch_uow_xda_p_reads - Skipped prefetch unit of work XDA data physical reads monitor element |
| SKIPPED_PREFETCH_UOW_TEMP_DATA_P_READS | BIGINT | skipped_prefetch_uow_temp_data_p_reads - Skipped prefetch unit of work temporary data physical reads monitor element |
| SKIPPED_PREFETCH_UOW_TEMP_INDEX_P_READS | BIGINT | skipped_prefetch_uow_temp_index_p_reads - Skipped prefetch unit of work temporary index physical reads monitor element |
| SKIPPED_PREFETCH_UOW_TEMP_XDA_P_READS | BIGINT | skipped_prefetch_uow_temp_xda_p_reads - Skipped prefetch unit of work temporary XDA data physical reads monitor element |
| POOL_ASYNC_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_async_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found by asynchronous EDUs in a local buffer pool monitor element |
| POOL_ASYNC_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_async_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found by asynchronous EDUs in a local buffer pool monitor element |
| POOL_ASYNC_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_async_xda_gbp_indep_pages_found_in_lbp - Group buffer pool independent XML storage object(XDA) pages found by asynchronous EDUs in a local buffer pool monitor element |
| CACHING_TIER | VARCHAR(8) | Reserved for future use. |
| CACHING_TIER_IO_ERRORS | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|---|-----------|--------------------------|
| POOL_COL_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_READ_TIME | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_WRITE_TIME | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_READS | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_READS | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_READS | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_READS | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|---|-----------|---|
| POOL_ASYNC_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted monitor element |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted monitor element |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated monitor element |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total Hash Joins monitor element |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total Hash Loops monitor element |
| HASH_JOIN_OVERFLOW | BIGINT | hash_join_overflows - Hash Join Overflows monitor element |
| HASH_JOIN_SMALL_OVERFLOW | BIGINT | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| POST_SHRTHRESHOLD_HASH_JOINS | BIGINT | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP Functions monitor element |
| OLAP_FUNC_OVERFLOW | BIGINT | olap_func_overflows - OLAP Function Overflows monitor element |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element |
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL Statements Attempted monitor element |
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed Statement Operations monitor element |
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL Statements Executed monitor element |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| MERGE_SQL_STMTS | BIGINT | merge_sql_stmts - Merge SQL statements executed |
| XQUERY_STMTS | BIGINT | xquery_stmts - XQuery Statements Attempted monitor element |
| IMPLICIT_REBINDS | BIGINT | implicit_rebinds - Number of implicit rebinds |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/Precompiles Attempted monitor element |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal Rows Inserted monitor element |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal Rows Updated monitor element |
| CALL_SQL_STMTS | BIGINT | call_sql_stmts - CALL SQL statements executed |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|---|-----------|---|
| POOL_COL_L_READS | BIGINT | pool_col_l_reads - Buffer pool column-organized logical reads |
| POOL_TEMP_COL_L_READS | BIGINT | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| POOL_COL_P_READS | BIGINT | pool_col_p_reads - Buffer pool column-organized physical reads |
| POOL_TEMP_COL_P_READS | BIGINT | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| POOL_COL_LBP_PAGES_FOUND | BIGINT | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found |
| POOL_COL_WRITES | BIGINT | pool_col_writes - Buffer pool column-organized writes |
| POOL_ASYNC_COL_READS | BIGINT | pool_async_col_reads - Buffer pool asynchronous column-organized reads monitor element |
| POOL_ASYNC_COL_READ_REQS | BIGINT | pool_async_col_read_reqs - Buffer pool asynchronous column-organized read requests monitor element |
| POOL_ASYNC_COL_WRITES | BIGINT | pool_async_col_writes - Buffer pool asynchronous column-organized writes monitor element |
| POOL_ASYNC_COL_LBP_PAGES_FOUND | BIGINT | pool_async_col_lbp_pages_found - Asynchronous local buffer pool column-organized pages found monitor element |
| POOL_COL_GBP_L_READS | BIGINT | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| POOL_COL_GBP_P_READS | BIGINT | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| POOL_COL_GBP_INVALID_PAGES | BIGINT | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |
| POOL_ASYNC_COL_GBP_L_READS | BIGINT | pool_async_col_gbp_l_reads - Asynchronous group buffer pool column-organized logical reads monitor element |
| POOL_ASYNC_COL_GBP_P_READS | BIGINT | pool_async_col_gbp_p_reads - Asynchronous group buffer pool column-organized physical reads monitor element |
| POOL_ASYNC_COL_GBP_INVALID_PAGES | BIGINT | pool_async_col_gbp_invalid_pages - Asynchronous group buffer pool column-organized invalid data pages monitor element |
| POOL_ASYNC_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_async_col_gbp_indep_pages_found_in_lbp - Asynchronous group buffer pool column-organized independent data pages found in local buffer pool monitor element |
| POOL_QUEUED_ASYNC_COL_REQS | BIGINT | pool_queued_async_col_reqs - Column-organized prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_REQS | BIGINT | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| POOL_QUEUED_ASYNC_COL_PAGES | BIGINT | pool_queued_async_col_pages - Column-organized page prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_PAGES | BIGINT | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| POOL_FAILED_ASYNC_COL_REQS | BIGINT | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| POOL_FAILED_ASYNC_TEMP_COL_REQS | BIGINT | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |
| SKIPPED_PREFETCH_COL_P_READS | BIGINT | skipped_prefetch_col_p_reads - Skipped prefetch column-organized physical reads monitor element |
| SKIPPED_PREFETCH_TEMP_COL_P_READS | BIGINT | skipped_prefetch_temp_col_p_reads - Skipped prefetch column-organized temporary physical reads monitor element |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|---------------------------------------|-----------|---|
| SKIPPED_PREFETCH_UOW_COL_P_READS | BIGINT | skipped_prefetch_uow_col_p_reads - Skipped prefetch unit of work column-organized physical reads monitor element |
| SKIPPED_PREFETCH_UOW_TEMP_COL_P_READS | BIGINT | skipped_prefetch_uow_temp_col_p_reads - Skipped prefetch unit of work column-organized temporary physical reads monitor element |
| TOTAL_COL_TIME | BIGINT | total_col_time - Total column-organized time |
| TOTAL_COL_PROC_TIME | BIGINT | total_col_proc_time - Total column-organized processing time |
| TOTAL_COL_EXECUTIONS | BIGINT | total_col_executions - Total column-organized executions |
| NUM_POOLED_AGENTS | BIGINT | num_pooled_agents - Number of pooled agents monitor element |
| POST_THRESHOLD_HASH_JOINS | BIGINT | post_threshold_hash_joins - Hash Join Threshold monitor element |
| PKG_CACHE_NUM_OVERFLOWS | BIGINT | pkg_cache_num_overflows - Package Cache Overflows |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog Cache Overflows |
| TOTAL_ASYNC_RUNSTATS | BIGINT | total_async_runstats - Total number of asynchronous RUNSTATS requests |
| STATS_CACHE_SIZE | BIGINT | stats_cache_size - Size of statistics cache |
| TOTAL_HASH_GRPBYs | BIGINT | total_hash_grpbys - Total hash group by operations |
| HASH_GRPBY_OVERFLOWs | BIGINT | hash_grpby_overflows - Hash group by overflows |
| POST_THRESHOLD_HASH_GRPBYs | BIGINT | post_threshold_hash_grpbys - Hash group by threshold |
| ACTIVE_HASH_GRPBYs | BIGINT | active_hash_grpbys - Active hash group by operations |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total Sort Heap Allocated |
| SORT_SHRHEAP_ALLOCATED | BIGINT | sort_shrheap_allocated - Sort Share Heap Currently Allocated |
| SORT_SHRHEAP_TOP | BIGINT | sort_shrheap_top - Sort share heap high watermark |
| POST_THRESHOLD_OLAP_FUNCs | BIGINT | post_threshold_olap_funcs - OLAP function threshold monitor element |
| POST_THRESHOLD_COL_VECTOR_CONSUMERS | BIGINT | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| TOTAL_COL_VECTOR_CONSUMERS | BIGINT | total_col_vector_consumers - Total columnar_vector consumers |
| ACTIVE_HASH_GRPBYs_TOP | BIGINT | active_hash_grpbys_top - Active hash Group By operations high watermark |
| ACTIVE_HASH_JOINS_TOP | BIGINT | active_hash_joins_top - Active hash joins operations high watermark |
| ACTIVE_OLAP_FUNCs_TOP | BIGINT | active_olap_funcs_top - Active OLAP functions operations high watermark |
| ACTIVE_PEAS | BIGINT | active_peas - Active partial early aggregations operations |
| ACTIVE_PEAS_TOP | BIGINT | active_peas_top - Active partial early aggregations operations high watermark |
| ACTIVE_PEDS | BIGINT | active_peds - Active partial early distinct operations |
| ACTIVE_PEDS_TOP | BIGINT | active_peds_top - Active partial early distinct operations high watermark |
| ACTIVE_SORT_CONSUMERS | BIGINT | active_sort_consumers - Active sort consumers |
| ACTIVE_SORT_CONSUMERS_TOP | BIGINT | active_sort_consumers_top - Active sort consumers high watermark |
| ACTIVE_SORTS_TOP | BIGINT | active_sorts_top - Active Sorts high watermark |
| ACTIVE_COL_VECTOR_CONSUMERS | BIGINT | active_col_vector_consumers - Active columnar_vector consumers |
| ACTIVE_COL_VECTOR_CONSUMERS_TOP | BIGINT | active_col_vector_consumers_top - Active columnar_vector consumers high watermark |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|------------------------------|-----------|--|
| SORT_CONSUMER_HEAP_TOP | BIGINT | sort_consumer_heap_top - Individual private sort heap consumer high watermark |
| SORT_CONSUMER_SHRHEAP_TOP | BIGINT | sort_consumer_shrheap_top - Individual shared sort heap consumer high watermark |
| SORT_HEAP_TOP | BIGINT | sort_heap_top - Sort private heap high watermark |
| TOTAL_BACKUP_TIME | BIGINT | total_backup_time - Total elapsed time for online backups |
| TOTAL_BACKUP_PROC_TIME | BIGINT | total_backup_proc_time - Total non-wait time for online backups |
| TOTAL_BACKUPS | BIGINT | total_backups - Total backups |
| TOTAL_INDEX_BUILD_TIME | BIGINT | total_index_build_time - Total time spent building indexes due to index creation |
| TOTAL_INDEX_BUILD_PROC_TIME | BIGINT | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation |
| TOTAL_INDEXES_BUILT | BIGINT | total_indexes_built - Total number of indexes built |
| IDA_SEND_WAIT_TIME | BIGINT | ida_send_wait_time - Time spent waiting to send data monitor element |
| IDA_SENDS_TOTAL | BIGINT | ida_sends_total - Number of times data sent monitor element |
| IDA_SEND_VOLUME | BIGINT | ida_send_volume - Total data volume sent monitor element |
| IDA_RECV_WAIT_TIME | BIGINT | ida_recv_wait_time - Time spent waiting to receive data monitor element |
| IDA_RECVS_TOTAL | BIGINT | ida_recv_total - Number of times data received monitor element |
| IDA_RECV_VOLUME | BIGINT | ida_recv_volume - Total data volume received monitor element |
| FCM_TQ_RECV_WAITS_TOTAL | BIGINT | fcm_tq_recv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| FCM_MESSAGE_RECV_WAITS_TOTAL | BIGINT | fcm_message_recv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| FCM_TQ_SEND_WAITS_TOTAL | BIGINT | fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |
| FCM_MESSAGE_SEND_WAITS_TOTAL | BIGINT | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| FCM_SEND_WAITS_TOTAL | BIGINT | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| FCM_RECV_WAITS_TOTAL | BIGINT | fcm_recv_waits_total - Number of times spent waiting to receive data through FCM monitor element |
| EXT_TABLE_RECV_WAIT_TIME | BIGINT | ext_table_recv_wait_time - Total agent wait time for external table readers monitor element |
| EXT_TABLE_RECVS_TOTAL | BIGINT | ext_table_recv_total - Total row batches received from external table readers monitor element |
| EXT_TABLE_RECV_VOLUME | BIGINT | ext_table_recv_volume - Total data received from external table readers monitor element |
| EXT_TABLE_READ_VOLUME | BIGINT | ext_table_read_volume - Total data read by external table readers monitor element |
| EXT_TABLE_SEND_WAIT_TIME | BIGINT | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| EXT_TABLE_SENDS_TOTAL | BIGINT | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| EXT_TABLE_SEND_VOLUME | BIGINT | ext_table_send_volume - Total data sent to external table writers monitor element |
| EXT_TABLE_WRITE_VOLUME | BIGINT | ext_table_write_volume - Total data written by external table writers monitor element |
| COL_VECTOR_CONSUMER_OVERFLOW | BIGINT | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |

Table 125. Information returned by the MON_GET_DATABASE routine (continued)

| Column name | Data type | Description |
|-------------------------------|-----------|--|
| TOTAL_COL_SYNOPSIS_TIME | BIGINT | total_col_synopsis_time - Total column-organized synopsis table time monitor element |
| TOTAL_COL_SYNOPSIS_PROC_TIME | BIGINT | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| TOTAL_COL_SYNOPSIS_EXECUTIONS | BIGINT | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |
| COL_SYNOPSIS_ROWS_INSERTED | BIGINT | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| LOB_PREFETCH_WAIT_TIME | BIGINT | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| LOB_PREFETCH_REQS | BIGINT | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| FED_ROWS_DELETED | BIGINT | fed_rows_deleted - Rows deleted by a federation system monitor element |
| FED_ROWS_INSERTED | BIGINT | fed_rows_inserted - Rows inserted by a federation system monitor element |
| FED_ROWS_UPDATED | BIGINT | fed_rows_updated - Rows updated by a federation system monitor element |
| FED_ROWS_READ | BIGINT | fed_rows_read - Rows read by a federation system monitor element |
| FED_WAIT_TIME | BIGINT | fed_wait_time - Time spent by a federation server monitor element |
| FED_WAITS_TOTAL | BIGINT | fed_waits_total - Total number of execution times for a federation server monitor element |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Insert count for SQL workspace monitor element |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - SQL workspace access count monitor element |
| SHR_WORKSPACE_ACTIVE | BIGINT | shr_workspace_active - Shared workspace active memory monitor element |
| SHR_WORKSPACE_INACTIVE | BIGINT | shr_workspace_inactive - Shared workspace inactive memory monitor element |
| ADM_OVERFLOWES | BIGINT | Reserved for future use. |
| ADM_BYPASS_ACT_TOTAL | BIGINT | Reserved for future use. |

Examples

1. Determine the activation time, activation state, total connections, and connection high water mark for the current database on all members:

```
SELECT DB_CONN_TIME, DB_ACTIVATION_STATE, TOTAL_CONS, CONNECTIONS_TOP
FROM TABLE (MON_GET_DATABASE(-2));
```

This query returns the following output:

| DB_CONN_TIME | DB_ACTIVATION_STATE | TOTAL_CONS | CONNECTIONS_TOP |
|----------------------------|---------------------|------------|-----------------|
| ----- | ----- | ----- | ----- |
| 2012-09-11-16.31.42.000000 | NONE | 23 | 13 |

2. Get the explicit hierarchical locking state for the database from all database members.

```
SELECT MEMBER,
       DATA_SHARING_REMOTE_LOCKWAIT_COUNT AS DSRL_COUNT,
       DATA_SHARING_REMOTE_LOCKWAIT_TIME AS DSRL_TIME
FROM TABLE (MON_GET_DATABASE(-2))
```

The query returns the following output, indicating that one table exited from a "NOT_SHARED" state, taking approximately 10 seconds to do so.

| MEMBER | DSRL_COUNT | DSRL_TIME |
|--------|------------|-----------|
| 0 | 0 | 0 |
| 1 | 1 | 10042 |
| 2 | 0 | 0 |

MON_GET_DATABASE_DETAILS table function - Get database metrics as an XML document

The MON_GET_DATABASE_DETAILS table function retrieves database metrics and returns the information in an XML document.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_GET_DATABASE_DETAILS—(—*member*—)—————►►

The schema is SYSPROC.

Routine parameters

member

An input parameter of type INTEGER that specifies a valid database member from which the records should be fetched. Specify -1 or null for the current member, or -2 for information from all active database members. An active database member a database that is available for connection and use by applications.

Information returned

Table 126. Information returned by the MON_GET_DATABASE_DETAILS routine

| Column name | Data type | Description |
|-------------|-----------|--|
| MEMBER | SMALLINT | member - Database member |
| DETAILS | BLOB(1M) | XML document that contains activity details. See Table 127 for a description of the elements in this document. |

Detailed metrics returned

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS

| Element name | Data type | Description or corresponding monitor element |
|-------------------|-----------|--|
| act_aborted_total | xs:long | act_aborted_total - Total aborted activities monitor element |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|---------------------------------|-----------------------|--|
| act_completed_total | xs:long | act_completed_total - Total completed activities monitor element |
| act_rejected_total | xs:long | act_rejected_total - Total rejected activities monitor element |
| act_rqsts_total | xs:long | act_rqsts_total - Total activity requests monitor elements |
| active_col_vector_consumers | xs:long | active_col_vector_consumers - Active columnar_vector consumers |
| active_col_vector_consumers_top | xs:long | active_col_vector_consumers_top - Active columnar_vector consumers high watermark |
| active_hash_grpbys | xs:long | active_hash_grpbys - Active hash group by operations |
| active_hash_grpbys_top | xs:long | active_hash_grpbys_top - Active hash Group By operations high watermark |
| active_hash_joins | xs:long | active_hash_joins - Active hash joins monitor element |
| active_hash_joins_top | xs:long | active_hash_joins_top - Active hash joins operations high watermark |
| active_olap_funcs | xs:long | active_olap_funcs - Active OLAP Functions monitor element |
| active_olap_funcs_top | xs:long | active_olap_funcs_top - Active OLAP functions operations high watermark |
| active_peas | xs:long | active_peas - Active partial early aggregations operations |
| active_peas_top | xs:long | active_peas_top - Active partial early aggregations operations high watermark |
| active_peds | xs:long | active_peds - Active partial early distinct operations |
| active_peds_top | xs:long | active_peds_top - Active partial early distinct operations high watermark |
| active_sort_consumers | xs:long | active_sort_consumers - Active sort consumers |
| active_sort_consumers_top | xs:long | active_sort_consumers_top - Active sort consumers high watermark |
| active_sorts | xs:long | active_sorts - Active Sorts monitor element |
| active_sorts_top | xs:long | active_sorts_top - Active Sorts high watermark |
| adm_bypass_act_total | xs:long | Reserved for future use. |
| adm_overflows | xs:long | Reserved for future use. |
| agent_wait_time | xs:long | agent_wait_time - Agent wait time monitor element |
| agent_waits_total | xs:long | agent_waits_total - Total agent waits monitor element |
| agents_top | xs:long | agents_top - Number of Agents Created monitor element |
| app_act_aborted_total | xs:long | app_act_aborted_total - Total failed external coordinator activities monitor element |
| app_act_completed_total | xs:long | app_act_completed_total - Total successful external coordinator activities monitor element |
| app_act_rejected_total | xs:long | app_act_rejected_total - Total rejected external coordinator activities monitor element |
| app_rqsts_completed_total | xs:long | app_rqsts_completed_total - Total application requests completed monitor element |
| appl_section_inserts | xs:nonNegativeInteger | appl_section_inserts - Insert count for SQL workspace monitor element |
| appl_section_lookups | xs:nonNegativeInteger | appl_section_lookups - SQL workspace access count monitor element |
| appls_cur_cons | xs:long | appls_cur_cons - Applications Connected Currently monitor element |
| appls_in_db2 | xs:long | appls_in_db2 - Applications Executing in the Database Currently monitor element |
| audit_events_total | xs:long | audit_events_total - Total audit events monitor element |
| audit_file_write_wait_time | xs:long | audit_file_write_wait_time - Audit file write wait time monitor element |
| audit_file_writes_total | xs:long | audit_file_writes_total - Total audit files written monitor element |
| audit_subsystem_wait_time | xs:long | audit_subsystem_wait_time - Audit subsystem wait time monitor element |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|------------------------------------|-----------------------|--|
| audit_subsystem_waits_total | xs:long | audit_subsystem_waits_total - Total audit subsystem waits monitor element |
| binds_precompiles | xs:long | binds_precompiles - Binds/Precompiles Attempted monitor element |
| block_ios | xs:long | block_ios - Number of block I/O requests monitor element |
| call_sql_stmts | xs:long | call_sql_stmts - CALL SQL statements executed |
| cat_cache_inserts | xs:long | cat_cache_inserts - Catalog cache inserts monitor element |
| cat_cache_lookups | xs:long | cat_cache_lookups - Catalog cache lookups monitor element |
| cat_cache_overflows | xs:long | cat_cache_overflows - Catalog Cache Overflows |
| catalog_partition | xs:short | catalog_node - Catalog Node Number |
| cf_wait_time | xs:long | cf_wait_time - cluster caching facility wait time monitor element |
| cf_waits | xs:long | cf_waits - Number of cluster caching facility waits monitor element |
| client_idle_wait_time | xs:long | client_idle_wait_time - Client idle wait time monitor element |
| col_synopsis_rows_inserted | xs:nonNegativeInteger | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| col_vector_consumer_overflows | xs:long | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |
| comm_exit_wait_time | xs:long | comm_exit_wait_time - Communication buffer exit wait time monitor element |
| comm_exit_waits | xs:long | comm_exit_waits - Communication buffer exit number of waits monitor element |
| connections_top | xs:long | connections_top - Maximum Number of Concurrent Connections monitor element |
| coord_agents_top | xs:long | coord_agents_top - Maximum Number of Coordinating Agents monitor element |
| data_sharing_remote_lockwait_count | xs:long | data_sharing_remote_lockwait_count - Data sharing remote lock wait count monitor element |
| data_sharing_remote_lockwait_time | xs:long | data_sharing_remote_lockwait_time - Data sharing remote lock wait time monitor element |
| db_activation_state | xs:db2_string_type_10 | db_activation_state - Database activation state |
| db_conn_time | xs:dateTime | db_conn_time - Database activation timestamp monitor element |
| db_path | xs:long | db_path - Database Path monitor element |
| db_status | xs:db2_string_type_16 | db_status - Status of database monitor element |
| ddl_sql_stmts | xs:long | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| deadlocks | xs:long | deadlocks - Deadlocks detected monitor element |
| diaglog_write_wait_time | xs:long | diaglog_write_wait_time - Diagnostic log file write wait time monitor element |
| diaglog_writes_total | xs:long | diaglog_writes_total - Total diagnostic log file writes monitor element |
| direct_read_reqs | xs:long | direct_read_reqs - Direct read requests monitor element |
| direct_read_time | xs:long | direct_read_time - Direct read time monitor element |
| direct_reads | xs:long | direct_reads - Direct reads from database monitor element |
| direct_write_reqs | xs:long | direct_write_reqs - Direct write requests monitor element |
| direct_write_time | xs:long | direct_write_time - Direct write time monitor element |
| direct_writes | xs:long | direct_writes - Direct writes to database monitor element |
| disabled_peds | xs:long | disabled_peds - Disabled partial early distincts monitor element |
| dynamic_sql_stmts | xs:long | dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element |
| evmon_wait_time | xs:long | evmon_wait_time - Event monitor wait time monitor element |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|------------------------------|-----------------------|--|
| evmon_waits_total | xs:long | evmon_waits_total - Event monitor total waits monitor element |
| ext_table_read_volume | xs:nonNegativeInteger | ext_table_read_volume - Total data read by external table readers monitor element |
| ext_table_rcv_volume | xs:nonNegativeInteger | ext_table_rcv_volume - Total data received from external table readers monitor element |
| ext_table_rcv_wait_time | xs:nonNegativeInteger | ext_table_rcv_wait_time - Total agent wait time for external table readers monitor element |
| ext_table_rcvs_total | xs:nonNegativeInteger | ext_table_rcvs_total - Total row batches received from external table readers monitor element |
| ext_table_send_volume | xs:nonNegativeInteger | ext_table_send_volume - Total data sent to external table writers monitor element |
| ext_table_send_wait_time | xs:nonNegativeInteger | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| ext_table_sends_total | xs:nonNegativeInteger | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| ext_table_write_volume | xs:nonNegativeInteger | ext_table_write_volume - Total data written by external table writers monitor element |
| failed_sql_stmts | xs:long | failed_sql_stmts - Failed Statement Operations monitor element |
| fcm_message_rcv_volume | xs:long | fcm_message_rcv_volume - FCM message received volume monitor element |
| fcm_message_rcv_wait_time | xs:long | fcm_message_rcv_wait_time - FCM message received wait time monitor element |
| fcm_message_rcv_waits_total | xs:nonNegativeInteger | fcm_message_rcv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| fcm_message_rcvs_total | xs:long | fcm_message_rcvs_total - Total FCM message receives monitor element |
| fcm_message_send_volume | xs:long | fcm_message_send_volume - FCM message send volume monitor element |
| fcm_message_send_wait_time | xs:long | fcm_message_send_wait_time - FCM message send wait time monitor element |
| fcm_message_send_waits_total | xs:nonNegativeInteger | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| fcm_message_sends_total | xs:long | fcm_message_sends_total - Total FCM message sends monitor element |
| fcm_rcv_volume | xs:long | fcm_rcv_volume - FCM received volume monitor element |
| fcm_rcv_wait_time | xs:long | fcm_rcv_wait_time - FCM received wait time monitor element |
| fcm_rcv_waits_total | xs:nonNegativeInteger | fcm_rcv_waits_total - Number of times spent waiting to receive data through FCM monitor element |
| fcm_rcvs_total | xs:long | fcm_rcvs_total - FCM receives total monitor element |
| fcm_send_volume | xs:long | fcm_send_volume - FCM send volume monitor element |
| fcm_send_wait_time | xs:long | fcm_send_wait_time - FCM send wait time monitor element |
| fcm_send_waits_total | xs:nonNegativeInteger | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| fcm_sends_total | xs:long | fcm_sends_total - FCM sends total monitor element |
| fcm_tq_rcv_volume | xs:long | fcm_tq_rcv_volume - FCM table queue received volume monitor element |
| fcm_tq_rcv_wait_time | xs:long | fcm_tq_rcv_wait_time - FCM table queue received wait time monitor element |
| fcm_tq_rcv_waits_total | xs:nonNegativeInteger | fcm_tq_rcv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| fcm_tq_rcvs_total | xs:long | fcm_tq_rcvs_total - FCM table queue receives total monitor element |
| fcm_tq_send_volume | xs:long | fcm_tq_send_volume - FCM table queue send volume monitor element |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|---------------------------|-----------------------|--|
| fcmtq_send_wait_time | xs:long | fcmtq_send_wait_time - FCM table queue send wait time monitor element |
| fcmtq_send_waits_total | xs:nonNegativeInteger | fcmtq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |
| fcmtq_sends_total | xs:long | fcmtq_sends_total - FCM table queue send total monitor element |
| fed_rows_deleted | xs:nonNegativeInteger | fed_rows_deleted - Rows deleted by a federation system monitor element |
| fed_rows_inserted | xs:nonNegativeInteger | fed_rows_inserted - Rows inserted by a federation system monitor element |
| fed_rows_read | xs:nonNegativeInteger | fed_rows_read - Rows read by a federation system monitor element |
| fed_rows_updated | xs:nonNegativeInteger | fed_rows_updated - Rows updated by a federation system monitor element |
| fed_wait_time | xs:nonNegativeInteger | fed_wait_time - Time spent by a federation server monitor element |
| fed_waits_total | xs:nonNegativeInteger | fed_waits_total - Total number of execution times for a federation server monitor element |
| files_closed | xs:long | files_closed - Database files closed monitor element |
| hash_grpby_overflows | xs:long | hash_grpby_overflows - Hash group by overflows |
| hash_join_overflows | xs:long | hash_join_overflows - Hash Join Overflows monitor element |
| hash_join_small_overflows | xs:long | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| ida_rcv_volume | xs:long | ida_rcv_volume - Total data volume received monitor element |
| ida_rcv_wait_time | xs:long | ida_rcv_wait_time - Time spent waiting to receive data monitor element |
| ida_rcvs_total | xs:long | ida_rcvs_total - Number of times data received monitor element |
| ida_send_volume | xs:long | ida_send_volume - Total data volume sent monitor element |
| ida_send_wait_time | xs:long | ida_send_wait_time - Time spent waiting to send data monitor element |
| ida_sends_total | xs:long | ida_sends_total - Number of times data sent monitor element |
| implicit_rebinds | xs:long | implicit_rebinds - Number of implicit rebinds |
| int_commits | xs:long | int_commits - Internal commits monitor element |
| int_rollbacks | xs:long | int_rollbacks - Internal rollbacks monitor element |
| int_rows_deleted | xs:long | int_rows_deleted - Internal Rows Deleted monitor element |
| int_rows_inserted | xs:long | int_rows_inserted - Internal Rows Inserted monitor element |
| int_rows_updated | xs:long | int_rows_updated - Internal Rows Updated monitor element |
| ipc_rcv_volume | xs:long | ipc_rcv_volume - Interprocess communication received volume monitor element |
| ipc_rcv_wait_time | xs:long | ipc_rcv_wait_time - Interprocess communication received wait time monitor element |
| ipc_rcvs_total | xs:long | ipc_rcvs_total - Interprocess communication receives total monitor element |
| ipc_send_volume | xs:long | ipc_send_volume - Interprocess communication send volume monitor element |
| ipc_send_wait_time | xs:long | ipc_send_wait_time - Interprocess communication send wait time monitor element |
| ipc_sends_total | xs:long | ipc_sends_total - Interprocess communication send total monitor element |
| last_backup | xs:dateTime | last_backup - Last Backup Timestamp monitor element |
| lob_prefetch_reqs | xs:nonNegativeInteger | lob_prefetch_reqs - LOB prefetcher request count monitor element |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|---|-----------------------|---|
| lob_prefetch_wait_time | xs:nonNegativeInteger | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| lock_escals | xs:long | lock_escals - Number of lock escalations monitor element |
| lock_escals_global | xs:long | lock_escals_global - Number of global lock escalations monitor element |
| lock_escals_locklist | xs:long | lock_escals_locklist - Number of locklist lock escalations monitor element |
| lock_escals_maxlocks | xs:long | lock_escals_maxlocks - Number of maxlocks lock escalations monitor element |
| lock_list_in_use | xs:long | lock_list_in_use - Total lock list memory in use monitor element |
| lock_timeouts | xs:long | lock_timeouts - Number of lock timeouts monitor element |
| lock_timeouts_global | xs:long | lock_timeouts_global - Lock timeouts global monitor element |
| lock_wait_time | xs:long | lock_wait_time - Time waited on locks monitor element |
| lock_wait_time_global | xs:long | lock_wait_time_global - Lock wait time global monitor element |
| lock_waits_global | xs:long | lock_waits_global - Lock waits global monitor element |
| log_buffer_wait_time | xs:long | log_buffer_wait_time - Log buffer wait time monitor element |
| log_disk_wait_time | xs:long | log_disk_wait_time - Log disk wait time monitor element |
| log_disk_waits_total | xs:long | log_disk_waits_total - Total log disk waits monitor element |
| member | xs:short | member - Database member monitor element |
| merge_sql_stmts | xs:long | merge_sql_stmts - Merge SQL statements executed |
| num_assoc_agents | xs:long | num_assoc_agents - Number of Associated Agents monitor element |
| num_coord_agents | xs:long | num_coord_agents - Number of coordinator agents |
| num_locks_held | xs:long | locks_held - Locks held monitor |
| num_locks_waiting | xs:long | lock_waits - Lock waits |
| num_log_buffer_full | xs:long | num_log_buffer_full - Number of times full log buffer caused agents to wait monitor element |
| num_lw_thresh_exceeded | xs:long | num_lw_thresh_exceeded - Number of lock wait thresholds exceeded monitor element |
| num_pooled_agents | xs:long | num_pooled_agents - Number of pooled agents monitor element |
| olap_func_overflows | xs:long | olap_func_overflows - OLAP Function Overflows monitor element |
| pages_from_block_ios | xs:long | pages_from_block_ios - Total number of pages read by block I/O monitor element |
| pages_from_vectored_ios | xs:long | pages_from_vectored_ios - Total number of pages read by vectored I/O monitor element |
| pkg_cache_inserts | xs:long | pkg_cache_inserts - Package cache inserts monitor element |
| pkg_cache_lookups | xs:long | pkg_cache_lookups - Package cache lookups monitor element |
| pkg_cache_num_overflows | xs:long | pkg_cache_num_overflows - Package Cache Overflows |
| pool_async_col_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_async_col_gbp_indep_pages_found_in_lbp - Asynchronous group buffer pool column-organized independent data pages found in local buffer pool monitor element |
| pool_async_col_gbp_invalid_pages | xs:nonNegativeInteger | pool_async_col_gbp_invalid_pages - Asynchronous group buffer pool column-organized invalid data pages monitor element |
| pool_async_col_gbp_l_reads | xs:nonNegativeInteger | pool_async_col_gbp_l_reads - Asynchronous group buffer pool column-organized logical reads monitor element |
| pool_async_col_gbp_p_reads | xs:nonNegativeInteger | pool_async_col_gbp_p_reads - Asynchronous group buffer pool column-organized physical reads monitor element |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|---|-----------------------|---|
| pool_async_col_lbp_pages_found | xs:nonNegativeInteger | pool_async_col_lbp_pages_found - Asynchronous local buffer pool column-organized pages found monitor element |
| pool_async_col_read_reqs | xs:nonNegativeInteger | pool_async_col_read_reqs - Buffer pool asynchronous column-organized read requests monitor element |
| pool_async_col_reads | xs:nonNegativeInteger | pool_async_col_reads - Buffer pool asynchronous column-organized reads monitor element |
| pool_async_col_writes | xs:nonNegativeInteger | pool_async_col_writes - Buffer pool asynchronous column-organized writes monitor element |
| pool_async_data_gbp_indep_pages_found_in_lbp | xs:long | pool_async_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found by asynchronous EDUs in a local buffer pool monitor element monitor element |
| pool_async_data_gbp_invalid_pages | xs:long | pool_async_data_gbp_invalid_pages - Asynchronous group buffer pool invalid data pages monitor element |
| pool_async_data_gbp_l_reads | xs:long | pool_async_data_gbp_l_reads - Asynchronous group buffer pool data logical reads monitor element |
| pool_async_data_gbp_p_reads | xs:long | pool_async_data_gbp_p_reads - Asynchronous group buffer pool data physical reads monitor element |
| pool_async_data_lbp_pages_found | xs:long | pool_async_data_lbp_pages_found - Asynchronous local buffer pool data pages found monitor element |
| pool_async_data_read_reqs | xs:long | pool_async_data_read_reqs - Buffer pool asynchronous read requests monitor element |
| pool_async_data_reads | xs:long | pool_async_data_reads - Buffer pool asynchronous data reads monitor element |
| pool_async_data_writes | xs:long | pool_async_data_writes - Buffer pool asynchronous data writes monitor element |
| pool_async_index_gbp_indep_pages_found_in_lbp | xs:long | pool_async_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found by asynchronous EDUs in a local buffer pool monitor element monitor element |
| pool_async_index_gbp_invalid_pages | xs:long | pool_async_index_gbp_invalid_pages - Asynchronous group buffer pool invalid index pages monitor element |
| pool_async_index_gbp_l_reads | xs:long | pool_async_index_gbp_l_reads - Asynchronous group buffer pool index logical reads monitor element |
| pool_async_index_gbp_p_reads | xs:long | pool_async_index_gbp_p_reads - Asynchronous group buffer pool index physical reads monitor element |
| pool_async_index_lbp_pages_found | xs:long | pool_async_index_lbp_pages_found - Asynchronous local buffer pool index pages found monitor element |
| pool_async_index_read_reqs | xs:long | pool_async_index_read_reqs - Buffer pool asynchronous index read requests monitor element |
| pool_async_index_reads | xs:long | pool_async_index_reads - Buffer pool asynchronous index reads monitor element |
| pool_async_index_writes | xs:long | pool_async_index_writes - Buffer pool asynchronous index writes monitor element |
| pool_async_read_time | xs:long | pool_async_read_time - Buffer Pool Asynchronous Read Time monitor element |
| pool_async_write_time | xs:long | pool_async_write_time - Buffer pool asynchronous write time monitor element |
| pool_async_xda_gbp_indep_pages_found_in_lbp | xs:long | pool_async_xda_gbp_indep_pages_found_in_lbp - Group buffer pool independent XML storage object(XDA) pages found by asynchronous EDUs in a local buffer pool monitor element monitor element |
| pool_async_xda_gbp_invalid_pages | xs:long | pool_async_xda_gbp_invalid_pages - Asynchronous group buffer pool invalid XDA data pages monitor element |
| pool_async_xda_gbp_l_reads | xs:long | pool_async_xda_gbp_l_reads - Group buffer pool XDA data asynchronous logical read requests monitor element |
| pool_async_xda_gbp_p_reads | xs:long | pool_async_xda_gbp_p_reads - Group buffer pool XDA data asynchronous physical read requests monitor element |
| pool_async_xda_lbp_pages_found | xs:long | pool_async_xda_lbp_pages_found - Asynchronous local buffer pool XDA data pages found monitor element |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|--|-----------------------|--|
| pool_async_xda_read_reqs | xs:long | pool_async_xda_read_reqs - Buffer pool asynchronous XDA read requests monitor element |
| pool_async_xda_reads | xs:long | pool_async_xda_reads - Buffer pool asynchronous XDA data reads monitor element |
| pool_async_xda_writes | xs:long | pool_async_xda_writes - Buffer pool asynchronous XDA data writes monitor element |
| pool_col_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |
| pool_col_gbp_invalid_pages | xs:nonNegativeInteger | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| pool_col_gbp_l_reads | xs:nonNegativeInteger | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| pool_col_gbp_p_reads | xs:nonNegativeInteger | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| pool_col_l_reads | xs:nonNegativeInteger | pool_col_l_reads - Buffer pool column-organized logical reads |
| pool_col_lbp_pages_found | xs:nonNegativeInteger | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found |
| pool_col_p_reads | xs:nonNegativeInteger | pool_col_p_reads - Buffer pool column-organized physical reads |
| pool_col_writes | xs:nonNegativeInteger | pool_col_writes - Buffer pool column-organized writes |
| pool_data_gbp_indep_pages_found_in_lbp | xs:long | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| pool_data_gbp_invalid_pages | xs:long | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages monitor element |
| pool_data_gbp_l_reads | xs:long | pool_data_gbp_l_reads - Group buffer pool data logical reads monitor element |
| pool_data_gbp_p_reads | xs:long | pool_data_gbp_p_reads - Group buffer pool data physical reads monitor element |
| pool_data_l_reads | xs:long | pool_data_l_reads - Buffer pool data logical reads monitor element |
| pool_data_lbp_pages_found | xs:long | pool_data_lbp_pages_found - Local buffer pool found data pages monitor element |
| pool_data_p_reads | xs:long | pool_data_p_reads - Buffer pool data physical reads monitor element |
| pool_data_writes | xs:long | pool_data_writes - Buffer pool data writes monitor element |
| pool_drty_pg_steal_clns | xs:long | pool_drty_pg_steal_clns - Buffer pool victim page cleaners triggered monitor element |
| pool_drty_pg_thrsh_clns | xs:long | pool_drty_pg_thrsh_clns - Buffer pool threshold cleaners triggered monitor element |
| pool_failed_async_col_reqs | xs:nonNegativeInteger | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| pool_failed_async_data_reqs | xs:long | pool_failed_async_data_reqs - Failed data prefetch requests monitor element |
| pool_failed_async_index_reqs | xs:long | pool_failed_async_index_reqs - Failed index prefetch requests monitor element |
| pool_failed_async_other_reqs | xs:long | pool_failed_async_other_reqs - Failed non-prefetch requests monitor element |
| pool_failed_async_temp_data_reqs | xs:long | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element |
| pool_failed_async_temp_index_reqs | xs:long | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element |
| pool_failed_async_temp_col_reqs | xs:nonNegativeInteger | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |
| pool_failed_async_temp_xda_reqs | xs:long | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element |
| pool_failed_async_xda_reqs | xs:long | pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|---|-----------------------|--|
| pool_index_gbp_indep_pages_found_in_lbp | xs:long | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| pool_index_gbp_invalid_pages | xs:long | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages monitor element |
| pool_index_gbp_l_reads | xs:long | pool_index_gbp_l_reads - Group buffer pool index logical reads monitor element |
| pool_index_gbp_p_reads | xs:long | pool_index_gbp_p_reads - Group buffer pool index physical reads monitor elements |
| pool_index_l_reads | xs:long | pool_index_l_reads - Buffer pool index logical reads monitor element |
| pool_index_lbp_pages_found | xs:long | pool_index_lbp_pages_found - Local buffer pool index pages found monitor element |
| pool_index_p_reads | xs:long | pool_index_p_reads - Buffer pool index physical reads monitor element |
| pool_index_writes | xs:long | pool_index_writes - Buffer pool index writes monitor element |
| pool_lsn_gap_clns | xs:long | pool_lsn_gap_clns - Buffer pool log space cleaners triggered monitor element |
| pool_no_victim_buffer | xs:long | pool_no_victim_buffer - Buffer pool no victim buffers monitor element |
| pool_queued_async_col_reqs | xs:nonNegativeInteger | pool_queued_async_col_reqs - Column-organized prefetch requests |
| pool_queued_async_data_pages | xs:long | pool_queued_async_data_pages - Data pages prefetch requests monitor element |
| pool_queued_async_data_reqs | xs:long | pool_queued_async_data_reqs - Data prefetch requests monitor element |
| pool_queued_async_index_pages | xs:long | pool_queued_async_index_pages - Index pages prefetch requests monitor element |
| pool_queued_async_index_reqs | xs:long | pool_queued_async_index_reqs - Index prefetch requests monitor element |
| pool_queued_async_col_pages | xs:nonNegativeInteger | pool_queued_async_col_pages - Column-organized page prefetch requests |
| pool_queued_async_temp_col_pages | xs:nonNegativeInteger | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| pool_queued_async_temp_col_reqs | xs:nonNegativeInteger | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| pool_queued_async_other_reqs | xs:long | pool_queued_async_other_reqs - Other requests handled by prefetchers monitor element |
| pool_queued_async_temp_data_pages | xs:long | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_data_reqs | xs:long | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_index_pages | xs:long | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_index_reqs | xs:long | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_xda_pages | xs:long | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_xda_reqs | xs:long | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element |
| pool_queued_async_xda_pages | xs:long | pool_queued_async_xda_pages - XDA pages prefetch requests monitor element |
| pool_queued_async_xda_reqs | xs:long | pool_queued_async_xda_reqs - XDA prefetch requests monitor element |
| pool_read_time | xs:long | pool_read_time - Total buffer pool physical read time monitor element |
| pool_temp_col_l_reads | xs:nonNegativeInteger | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|---------------------------------------|-----------------------|--|
| pool_temp_col_p_reads | xs:nonNegativeInteger | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| pool_temp_data_l_reads | xs:long | pool_temp_data_l_reads - Buffer pool temporary data logical reads monitor element |
| pool_temp_data_p_reads | xs:long | pool_temp_data_p_reads - Buffer pool temporary data physical reads monitor element |
| pool_temp_index_l_reads | xs:long | pool_temp_index_l_reads - Buffer pool temporary index logical reads monitor element |
| pool_temp_index_p_reads | xs:long | pool_temp_index_p_reads - Buffer pool temporary index physical reads monitor element |
| pool_temp_xda_l_reads | xs:long | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads monitor element |
| pool_temp_xda_p_reads | xs:long | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads monitor element |
| pool_write_time | xs:long | pool_write_time - Total buffer pool physical write time monitor element |
| pool_xda_gbp_indep_pages_found_in_lbp | xs:long | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |
| pool_xda_gbp_invalid_pages | xs:long | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages monitor element |
| pool_xda_gbp_l_reads | xs:long | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests monitor element |
| pool_xda_gbp_p_reads | xs:long | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests monitor element |
| pool_xda_l_reads | xs:long | pool_xda_l_reads - Buffer pool XDA data logical reads monitor element |
| pool_xda_lbp_pages_found | xs:long | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found monitor element |
| pool_xda_p_reads | xs:long | pool_xda_p_reads - Buffer pool XDA data physical reads monitor element |
| pool_xda_writes | xs:long | pool_xda_writes - Buffer pool XDA data writes monitor element |
| post_shrthreshold_hash_joins | xs:long | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| post_shrthreshold_sorts | xs:long | post_shrthreshold_sorts - Post shared threshold sorts monitor element |
| post_threshold_col_vector_consumers | xs:long | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| post_threshold_hash_grpbys | xs:long | post_threshold_hash_grpbys - Hash group by threshold monitor element |
| post_threshold_hash_joins | xs:long | post_threshold_hash_joins - Hash Join Threshold monitor element |
| post_threshold_olap_funcs | xs:long | post_threshold_olap_funcs - OLAP function threshold monitor element |
| post_threshold_peas | xs:long | post_threshold_peas - Partial early aggregation threshold monitor element |
| post_threshold_peds | xs:long | post_threshold_peds - Partial early distincts threshold monitor element |
| post_threshold_sorts | xs:long | post_threshold_sorts - Post threshold sorts monitor element |
| prefetch_wait_time | xs:long | prefetch_wait_time - Time waited for prefetch monitor element |
| prefetch_waits | xs:long | prefetch_waits - Prefetcher wait count monitor element |
| locks_waiting | xs:long | locks_waiting - Current agents waiting on locks monitor element |
| reclaim_wait_time | xs:long | reclaim_wait_time - Reclaim wait time monitor element |
| rows_deleted | xs:long | rows_deleted - Rows deleted monitor element |
| rows_inserted | xs:long | rows_inserted - Rows inserted monitor element |
| rows_modified | xs:long | rows_modified - Rows modified monitor element |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|---|-----------------------|---|
| rows_read | xs:long | rows_read - Rows read monitor element |
| rows_returned | xs:long | rows_returned - Rows returned monitor element |
| rows_updated | xs:long | rows_updated - Rows updated monitor element |
| rqsts_completed_total | xs:long | rqsts_completed_total - Total requests completed monitor element |
| select_sql_stmts | xs:long | select_sql_stmts - Select SQL Statements Executed monitor element |
| shr_workspace_active | xs:nonNegativeInteger | shr_workspace_active - Shared workspace active memory monitor element |
| shr_workspace_inactive | xs:nonNegativeInteger | shr_workspace_inactive - Shared workspace inactive memory monitor element |
| skipped_prefetch_data_p_reads | xs:long | skipped_prefetch_data_p_reads - Skipped prefetch data physical reads monitor element |
| skipped_prefetch_index_p_reads | xs:long | skipped_prefetch_index_p_reads - Skipped prefetch index physical reads monitor element |
| skipped_prefetch_col_p_reads | xs:nonNegativeInteger | skipped_prefetch_col_p_reads - Skipped prefetch column-organized physical reads monitor element |
| skipped_prefetch_temp_col_p_reads | xs:nonNegativeInteger | skipped_prefetch_temp_col_p_reads - Skipped prefetch column-organized temporary physical reads monitor element |
| skipped_prefetch_uow_col_p_reads | xs:nonNegativeInteger | skipped_prefetch_uow_col_p_reads - Skipped prefetch unit of work column-organized physical reads monitor element |
| skipped_prefetch_uow_temp_col_p_reads | xs:nonNegativeInteger | skipped_prefetch_uow_temp_col_p_reads - Skipped prefetch unit of work column-organized temporary physical reads monitor element |
| skipped_prefetch_temp_data_p_reads | xs:long | skipped_prefetch_temp_data_p_reads - Skipped prefetch temporary data physical reads monitor element |
| skipped_prefetch_temp_index_p_reads | xs:long | skipped_prefetch_temp_index_p_reads - Skipped prefetch temporary index physical reads monitor element |
| skipped_prefetch_temp_xda_p_reads | xs:long | skipped_prefetch_temp_xda_p_reads - Skipped prefetch temporary XDA data physical reads monitor element |
| skipped_prefetch_uow_data_p_reads | xs:long | skipped_prefetch_uow_data_p_reads - Skipped prefetch unit of work data physical reads monitor element |
| skipped_prefetch_uow_index_p_reads | xs:long | skipped_prefetch_uow_index_p_reads - Skipped prefetch unit of work index physical reads monitor element |
| skipped_prefetch_uow_temp_data_p_reads | xs:long | skipped_prefetch_uow_temp_data_p_reads - Skipped prefetch unit of work temporary data physical reads monitor element |
| skipped_prefetch_uow_temp_index_p_reads | xs:long | skipped_prefetch_uow_temp_index_p_reads - Skipped prefetch unit of work temporary index physical reads monitor element |
| skipped_prefetch_uow_temp_xda_p_reads | xs:long | skipped_prefetch_uow_temp_xda_p_reads - Skipped prefetch unit of work temporary XDA data physical reads monitor element |
| skipped_prefetch_uow_xda_p_reads | xs:long | skipped_prefetch_uow_xda_p_reads - Skipped prefetch unit of work XDA data physical reads monitor element |
| skipped_prefetch_xda_p_reads | xs:long | skipped_prefetch_xda_p_reads - Skipped prefetch XDA physical reads monitor element |
| sort_consumer_heap_top | xs:long | sort_consumer_heap_top - Individual private sort heap consumer high watermark |
| sort_consumer_shrheap_top | xs:long | sort_consumer_shrheap_top - Individual shared sort heap consumer high watermark |
| sort_heap_allocated | xs:long | sort_heap_allocated - Total Sort Heap Allocated monitor element |
| sort_heap_top | xs:long | sort_heap_top - Sort private heap high watermark |
| sort_overflows | xs:long | sort_overflows - Sort overflows monitor element |
| sort_shrheap_allocated | xs:long | sort_shrheap_allocated - Sort Share Heap Currently Allocated monitor element |
| sort_shrheap_top | xs:long | sort_shrheap_top - Sort share heap high watermark monitor element |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|--|-----------------------|---|
| spacemappage_reclaim_wait_time | xs:long | spacemappage_reclaim_wait_time - Space map page reclaim wait time monitor element |
| static_sql_stmts | xs:long | static_sql_stmts - Static SQL Statements Attempted monitor element |
| stats_cache_size | xs:long | stats_cache_size - Size of statistics cache |
| tcpip_rcv_volume | xs:long | tcpip_rcv_volume - TCP/IP received volume monitor element |
| tcpip_rcv_wait_time | xs:long | tcpip_rcv_wait_time - TCP/IP received wait time monitor element |
| tcpip_rcvs_total | xs:long | tcpip_rcvs_total - TCP/IP receives total monitor element |
| tcpip_send_volume | xs:long | tcpip_send_volume - TCP/IP send volume monitor element |
| tcpip_send_wait_time | xs:long | tcpip_send_wait_time - TCP/IP send wait time monitor element |
| tcpip_sends_total | xs:long | tcpip_sends_total - TCP/IP sends total monitor element |
| thresh_violations | xs:long | thresh_violations - Number of threshold violations monitor element |
| total_act_time | xs:long | total_act_time - Total activity time monitor element |
| total_act_wait_time | xs:long | total_act_wait_time - Total activity wait time monitor element |
| total_app_commits | xs:long | total_app_commits - Total application commits monitor elements |
| total_app_rollbacks | xs:long | total_app_rollbacks - Total application rollbacks monitor element |
| total_app_rqst_time | xs:long | total_app_rqst_time - Total application request time monitor element |
| total_app_section_executions | xs:long | total_app_section_executions - Total application section executions monitor element |
| total_async_runstats | xs:long | total_async_runstats - Total number of asynchronous RUNSTATS requests |
| total_backup_proc_time | xs:long | total_backup_proc_time - Total non-wait time for online backups |
| total_backup_time | xs:long | total_backup_time - Total elapsed time for online backups |
| total_backups | xs:long | total_backups - Total backups |
| total_col_executions | xs:nonNegativeInteger | total_col_executions - Total column-organized executions |
| total_col_proc_time | xs:nonNegativeInteger | total_col_proc_time - Total column-organized processing time |
| total_col_synopsis_executions | xs:nonNegativeInteger | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |
| total_col_synopsis_proc_time | xs:nonNegativeInteger | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| total_col_synopsis_time | xs:nonNegativeInteger | total_col_synopsis_time - Total column-organized synopsis table time monitor element |
| total_col_time | xs:nonNegativeInteger | total_col_time - Total column-organized time |
| total_col_vector_consumers | xs:long | total_col_vector_consumers - Total columnar_vector consumers |
| total_commit_proc_time | xs:long | total_commit_proc_time - Total commits processing time monitor element |
| total_commit_time | xs:long | total_commit_time - Total commit time monitor element |
| total_compilations | xs:long | total_compilations - Total compilations monitor element |
| total_compile_proc_time | xs:long | total_compile_proc_time - Total compile processing time monitor element |
| total_compile_time | xs:long | total_compile_time - Total compile time monitor element |
| total_connect_authentication_proc_time | xs:long | total_connect_authentication_proc_time - Total connection authentication processing time monitor element |
| total_connect_authentication_time | xs:long | total_connect_authentication_time - Total connection or switch user authentication request time monitor element |
| total_connect_authentications | xs:long | total_connect_authentications - Connections or switch user authentications performed monitor element |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|-----------------------------------|-----------|---|
| total_connect_request_proc_time | xs:long | total_connect_request_proc_time - Total connection or switch user request processing time monitor element |
| total_connect_request_time | xs:long | total_connect_request_time - Total connection or switch user request time monitor element |
| total_connect_requests | xs:long | total_connect_requests - Connection or switch user requests monitor element |
| total_cons | xs:long | total_cons - Connects Since Database Activation monitor element |
| total_cpu_time | xs:long | total_cpu_time - Total CPU time monitor element |
| total_disp_run_queue_time | xs:long | total_disp_run_queue_time - Total dispatcher run queue time monitor element |
| total_extended_latch_wait_time | xs:long | total_extended_latch_wait_time - Total extended latch wait time monitor element |
| total_extended_latch_waits | xs:long | total_extended_latch_waits - Total extended latch waits monitor element |
| total_hash_grpbys | xs:long | total_hash_grpbys - Total hash group by operations |
| total_hash_joins | xs:long | total_hash_joins - Total Hash Joins monitor element |
| total_hash_loops | xs:long | total_hash_loops - Total Hash Loops monitor element |
| total_implicit_compilations | xs:long | total_implicit_compilations - Total implicit complications monitor element |
| total_implicit_compile_proc_time | xs:long | total_implicit_compile_proc_time - Total implicit compile processing time monitor element |
| total_implicit_compile_time | xs:long | total_implicit_compile_time - Total implicit compile time monitor element |
| total_index_build_proc_time | xs:long | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation |
| total_index_build_time | xs:long | total_index_build_time - Total time spent building indexes due to index creation |
| total_indexes_built | xs:long | total_indexes_built - Total number of indexes built |
| total_load_proc_time | xs:long | total_load_proc_time - Total load processing time monitor element |
| total_load_time | xs:long | total_load_time - Total load time monitor element |
| total_loads | xs:long | total_loads - Total loads monitor element |
| total_olap_funcs | xs:long | total_olap_funcs - Total OLAP Functions monitor element |
| total_peas | xs:long | total_peas - Total partial early aggregations monitor element |
| total_peds | xs:long | total_peds - Total partial early distincts monitor element |
| total_reorg_proc_time | xs:long | total_reorg_proc_time - Total reorganization processing time monitor element |
| total_reorg_time | xs:long | total_reorg_time - Total reorganization time monitor element |
| total_reorgs | xs:long | total_reorgs - Total reorganizations monitor element |
| total_rollback_proc_time | xs:long | total_rollback_proc_time - Total rollback processing time monitor element |
| total_rollback_time | xs:long | total_rollback_time - Total rollback time monitor element |
| total_routine_invocations | xs:long | total_routine_invocations - Total routine invocations monitor elements |
| total_routine_time | xs:long | total_routine_time - Total routine time monitor element |
| total_routine_user_code_proc_time | xs:long | total_routine_user_code_proc_time - Total routine user code processing time monitor element |
| total_routine_user_code_time | xs:long | total_routine_user_code_time - Total routine user code time monitor element |
| total_rqst_mapped_in | xs:long | total_rqst_mapped_in - Total request mapped-in monitor element |
| total_rqst_mapped_out | xs:long | total_rqst_mapped_out - Total request mapped-out monitor element |
| total_rqst_time | xs:long | total_rqst_time - Total request time monitor element |

Table 127. Detailed metrics returned for MON_GET_DATABASE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|-----------------------------------|-----------|--|
| total_runstats | xs:long | total_runstats - Total runtime statistics monitor element |
| total_runstats_proc_time | xs:long | total_runstats_proc_time - Total runtime statistics processing time monitor element |
| total_runstats_time | xs:long | total_runstats_time - Total runtime statistics time monitor element |
| total_sec_cons | xs:long | total_sec_cons - Secondary Connections monitor element |
| total_section_proc_time | xs:long | total_section_proc_time - Total section processing time monitor element |
| total_section_sort_proc_time | xs:long | total_section_sort_proc_time - Total section sort processing time monitor element |
| total_section_sort_time | xs:long | total_section_sort_time - Total section sort time monitor element |
| total_section_sorts | xs:long | total_section_sorts - Total section sorts monitor element |
| total_section_time | xs:long | total_section_time - Total section time monitor element |
| total_sorts | xs:long | total_sorts - Total sorts monitor element |
| total_stats_fabrication_proc_time | xs:long | total_stats_fabrication_proc_time - Total statistics fabrication processing time monitor element |
| total_stats_fabrication_time | xs:long | total_stats_fabrication_time - Total statistics fabrication time monitor element |
| total_stats_fabrications | xs:long | total_stats_fabrications - Total statistics fabrications monitor elements |
| total_sync_runstats | xs:long | total_sync_runstats - Total synchronous RUNSTATS activities monitor element |
| total_sync_runstats_proc_time | xs:long | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time monitor element |
| total_sync_runstats_time | xs:long | total_sync_runstats_time - Total synchronous RUNSTATS time monitor elements |
| total_wait_time | xs:long | total_wait_time - Total wait time monitor element |
| tq_sort_heap_rejections | xs:long | tq_sort_heap_rejections - Table queue sort heap rejections monitor element |
| tq_sort_heap_requests | xs:long | tq_sort_heap_requests - Table queue sort heap requests monitor element |
| tq_tot_send_spills | xs:long | tq_tot_send_spills - Total number of table queue buffers overflowed monitor element |
| uid_sql_stmts | xs:long | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| unread_prefetch_pages | xs:long | unread_prefetch_pages - Unread prefetch pages monitor element |
| vectored_ios | xs:long | vectored_ios - Number of vectored I/O requests monitor element |
| wlm_queue_assignments_total | xs:long | wlm_queue_assignments_total - Workload manager total queue assignments monitor element |
| wlm_queue_time_total | xs:long | wlm_queue_time_total - Workload manager total queue time monitor element |
| xquery_stmts | xs:long | xquery_stmts - XQuery Statements Attempted monitor element |

Usage notes

The schema for the XML document that is returned in the DETAILS column is available in the file `sqllib/misc/DB2MonRoutines.xsd`. Further details can be found in the file `sqllib/misc/DB2MonCommon.xsd`.

Example

Retrieve information about `rows_read`, `num_locks_waiting`, `total_cpu_time`, and `direct_reads`, at the database level on the current member.

```

SELECT wlmetrics.member as member,
       detmetrics.rows_read as rows_read,
       detmetrics.num_locks_waiting as num_locks_waiting,
       detmetrics.total_cpu_time as total_cpu_time,
       detmetrics.direct_reads as direct_reads
FROM TABLE(MON_GET_DATABASE_DETAILS(NULL)) AS WLMETRICS,
XMLTABLE (XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'),
          '$detmetric/db2_database'
          PASSING XMLPARSE(DOCUMENT WLMETRICS.DETAILS) as "detmetric"
COLUMNS "ROWS_READ" INTEGER PATH 'system_metrics/rows_read',
         "NUM_LOCKS_WAITING" INTEGER PATH 'system_metrics/lock_waits',
         "TOTAL_CPU_TIME" INTEGER PATH 'system_metrics/total_cpu_time',
         "DIRECT_READS" INTEGER PATH 'system_metrics/direct_reads'
) AS DETMETRICS;

```

This query returns the following output:

| MEMBER | ROWS_READ | NUM_LOCKS_WAITING | TOTAL_CPU_TIME | DIRECT_READS |
|--------|-----------|-------------------|----------------|--------------|
| 0 | 3288 | 0 | 2330014 | - |

MON_GET_EXTENDED_LATCH_WAIT table function - Return information for latches

This function returns information for latches which have been involved in extended latch waits. This information includes the member, latch name, number of extended waits, and time spent in extended waits.

Syntax

►►—MON_GET_EXTENDED_LATCH_WAIT—(—*member*—)—————►►

The schema is SYSPROC.

Table function parameters

member

The member ID for which to return data. If NULL or -1 is specified, data for the currently connected member will be returned. If -2 is specified, data from all active members will be returned. Otherwise, only data for the specified member will be returned. If an invalid, undefined, or offline member is specified, no data will be returned.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Usage notes

The MON_GET_EXTENDED_LATCH_WAIT returns informations about which latches on a given member have been involved in extended latch waits, including

how much time and how many times the particular latch has been involved in extended latch waits. This information will be collected as part of the base monitor metrics.

Information returned

Table 128. Information returned for MON_GET_EXTENDED_LATCH_WAIT

| Column Name | Data Type | Description |
|--------------------------------|--------------|---|
| MEMBER | SMALLINT | member - Database member |
| LATCH_NAME | VARCHAR(256) | latch_name - Latch name monitor element |
| TOTAL_EXTENDED_LATCH_WAITS | BIGINT | total_extended_latch_waits - Total extended latch waits |
| TOTAL_EXTENDED_LATCH_WAIT_TIME | BIGINT | total_extended_latch_wait_time - Total extended latch wait time |

MON_GET_EXTENT_MOVEMENT_STATUS - get extent movement progress

The MON_GET_EXTENT_MOVEMENT_STATUS table function returns the status of the extent movement operation.

Syntax

►►—MON_GET_EXTENT_MOVEMENT_STATUS—(—*tbody_name*—,—*member*—)—►►

The schema is SYSPROC.

Table function parameters

tbody_name

An input argument of type VARCHAR(128) that specifies the table space to query. If the argument value is null, the function returns information for all table spaces.

member

An input argument of type INTEGER that specifies a valid member inside the same instance as the currently connected database. Specify -1 for the current database member, or -2 for all active database members. If the argument value is null, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

Retrieve all information about the current extent progress for all table spaces:

```
SELECT * FROM TABLE(SYSPROC.MON_GET_EXTENT_MOVEMENT_STATUS('', -1))
```

Here is an example of the output from the preceding query:

| TBSP_NAME | TBSP_ID | MEMBER | CURRENT_EXTENT | LAST_EXTENT | NUM_EXTENTS_MOVED |
|--------------|---------|--------|----------------|-------------|-------------------|
| SYSCATSPACE | 0 | 0 | -1 | -1 | -1 |
| TEMPSPACE1 | 1 | 0 | -1 | -1 | -1 |
| USERSPACE1 | 2 | 0 | -1 | -1 | -1 |
| TS1 | 3 | 0 | 1 | 2 | 3 |
| SYSTOOLSPACE | 4 | 0 | -1 | -1 | -1 |

5 record(s) selected.

Output from the query continued:

| ... | NUM_EXTENTS_LEFT | TOTAL_MOVE_TIME | ADDITIONAL_DETAILS |
|-----|------------------|-----------------|--------------------|
| ... | -1 | -1 | - |
| ... | -1 | -1 | - |
| ... | -1 | -1 | - |
| ... | 4 | 0 | - |
| ... | -1 | -1 | - |

Information returned

Table 129. Information returned for MON_GET_EXTENT_MOVEMENT_STATUS

| Column Name | Data Type | Description or corresponding monitor element |
|-------------------|--------------|--|
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TBSP_ID | BIGINT | tablespace_id - Table space identifier |
| MEMBER | SMALLINT | member - Member from which this information was collected |
| CURRENT_EXTENT | INTEGER | current_extent - Current extent being moved |
| LAST_EXTENT | INTEGER | last_extent - Last extent moved |
| NUM_EXTENTS_MOVED | INTEGER | num_extents_moved - Number of extents moved so far during this extent movement operation |
| NUM_EXTENTS_LEFT | INTEGER | num_extents_left - Number of extents left to move during this extent movement operation |
| TOTAL_MOVE_TIME | BIGINT | total_move_time - Total move time for all extents moved (in milliseconds) |

MON_GET_FCM - Get FCM metrics

The MON_GET_FCM table function returns metrics for the fast communication manager (FCM).

Syntax

►►—MON_GET_FCM—(—member—)—————◄◄

The schema is SYSPROC.

Table function parameter

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for information from all active database members. An active database member is where the database is available for connection and use by applications.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

To retrieve information about the fast communication manager message buffers on all members:

```
SELECT member, buff_free, buff_free_bottom
FROM TABLE (MON_GET_FCM (-2))
```

This query returns the following:

| MEMBER | BUFF_FREE | BUFF_FREE_BOTTOM |
|--------|-----------|------------------|
| 2 | 13425 | 13416 |
| 10 | 13425 | 13416 |
| 1 | 13425 | 13416 |

3 record(s) selected.

Information returned

Table 130. Information returned for MON_GET_FCM

| Column Name | Data Type | Description or corresponding monitor element |
|------------------|--------------|--|
| HOSTNAME | VARCHAR(128) | hostname - Host name |
| MEMBER | SMALLINT | member - Database member |
| BUFF_MAX | BIGINT | buff_max - Maximum possible number of FCM buffers |
| BUFF_TOTAL | BIGINT | buff_total - Number of currently allocated FCM buffers |
| BUFF_FREE | BIGINT | buff_free - FCM buffers currently free |
| BUFF_FREE_BOTTOM | BIGINT | buff_free_bottom - Minimum FCM buffers free |
| BUFF_AUTO_TUNING | SMALLINT | buff_auto_tuning - FCM buffer auto-tuning indicator |
| CH_MAX | BIGINT | ch_max - Maximum possible number of FCM channels |
| CH_TOTAL | BIGINT | ch_total - Number of currently allocated FCM channels |
| CH_FREE | BIGINT | ch_free - Channels currently free |

Table 130. Information returned for MON_GET_FCM (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|----------------|-----------|--|
| CH_FREE_BOTTOM | BIGINT | ch_free_bottom - Minimum channels free |
| CH_AUTO_TUNING | SMALLINT | ch_auto_tuning - FCM channel auto-tuning indicator |

Note: The metrics provided by this table function apply to all members on a given host machine. All members on a given host machine share the same set of buffers and channels. This means that the individual metrics will usually be the same for each member on given host machine. However, each member executes independently and the metrics might differ slightly as the resource numbers change in between the sampling on different members.

MON_GET_FCM_CONNECTION_LIST - Get details for all FCM connections

The MON_GET_FCM_CONNECTION_LIST table function returns monitor metrics for all the fast communication manager (FCM) connections on the specified member or members.

Syntax

►►—MON_GET_FCM_CONNECTION_LIST—(—*member*—)—————►►

The schema is SYSPROC.

Table function parameter

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for information from all active database members. An active database member is where the database is available for connection and use by applications.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Information returned

Table 131. Information returned for MON_GET_FCM_CONNECTION_LIST

| Column Name | Data Type | Description or corresponding monitor element |
|-------------------|-------------|--|
| MEMBER | SMALLINT | member - Database member |
| REMOTE_MEMBER | SMALLINT | remote_member - Remote member |
| CONNECTION_STATUS | VARCHAR(16) | connection_status - Connection status |

Table 131. Information returned for MON_GET_FCM_CONNECTION_LIST (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|-----------------------------|-----------|---|
| TOTAL_BUFFERS_SENT | BIGINT | total_buffers_sent - Total FCM buffers sent |
| TOTAL_BUFFERS_RCVD | BIGINT | total_buffers_rcvd - Total FCM buffers received |
| FCM_CONGESTION_TIME | BIGINT | fcm_congestion_time - FCM congestion time |
| FCM_CONGESTED_SENDS | BIGINT | fcm_congested_sends - FCM congested sends |
| FCM_NUM_CONGESTION_TIMEOUTS | BIGINT | fcm_num_congestion_timeouts - FCM congestion timeouts |
| FCM_SEND_VOLUME | BIGINT | fcm_send_volume - FCM send volume |
| FCM_RECV_VOLUME | BIGINT | fcm_rcv_volume - FCM received volume |
| FCM_MESSAGE_SEND_VOLUME | BIGINT | fcm_message_send_volume - FCM message send volume |
| FCM_MESSAGE_RECV_VOLUME | BIGINT | fcm_message_rcv_volume - FCM message received volume |
| FCM_TQ_SEND_VOLUME | BIGINT | fcm_tq_send_volume - FCM table queue send volume |
| FCM_TQ_RECV_VOLUME | BIGINT | fcm_tq_rcv_volume - FCM table queue received volume |
| FCM_NUM_CONN_LOST | BIGINT | fcm_num_conn_lost - FCM lost connections |
| FCM_NUM_CONN_TIMEOUTS | BIGINT | fcm_num_conn_timeouts - FCM connection timeouts |

MON_GET_GROUP_BUFFERPOOL

The MON_GET_GROUP_BUFFERPOOL table function returns statistics about the group buffer pool (GBP).

Syntax

►►—MON_GET_GROUP_BUFFERPOOL—(—*member*—)—————►►

Table function parameters

The schema is SYSPROC.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database. Specify -1 for the current database member, or -2 for all active members. If the NULL value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

If the group buffer pool (GBP) does not have sufficient space when attempting to register a page or write a page to the GBP, a GBP_FULL error occurs.

The following example returns the number of times the GBP_FULL error is encountered from all members.

```
SELECT SUM(T.NUM_GBP_FULL) AS NUM_GBP_FULL
FROM TABLE(MON_GET_GROUP_BUFFERPOOL(-2)) AS T
```

The following is an example of output from this query.

```
NUM_GBP_FULL
-----
          123
```

1 record(s) selected.

If the value of NUM_GBP_FULL increases by more than one per minute, then the current size of the GBP likely does not meet your needs. In this case, increase the size of the GBP with the command:

```
UPDATE DB CFG USING CF_GBP_SIZE <new_size>
```

For this command, the value of <new_size> grows the group buffer pool to a size sufficient to slow or stop the increasing number of GBP_FULL errors.

Information Returned

Table 132. Information returned for MON_GET_GROUP_BUFFERPOOL

| Column name | Data type | Description or corresponding monitor element |
|--------------|-----------|--|
| MEMBER | SMALLINT | member - Database member |
| NUM_GBP_FULL | BIGINT | Number of times the GBP_FULL error occurs. |

MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information

This function returns high availability disaster recovery (HADR) monitoring information.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_GET_HADR—(—member—)—————►►

The schema is SYSPROC.

Routine parameters

member

An input argument of type INTEGER that specifies a member number.
Returned rows represent log streams being processed by the member. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

Information returned

Table 133. Information returned for MON_GET_HADR

| Column Name | Data Type | Description |
|--------------------------|--------------|---|
| HADR_ROLE | VARCHAR(13) | hadr_role - HADR Role monitor element |
| REPLAY_TYPE | VARCHAR(9) | replay_type - HADR replication type monitor element |
| HADR_SYNCMODE | VARCHAR(10) | hadr_syncmode - HADR Synchronization Mode monitor element |
| STANDBY_ID | SMALLINT | standby_id - Standby ID monitor element |
| LOG_STREAM_ID | INTEGER | log_stream_id - Log stream ID monitor element |
| HADR_STATE | VARCHAR(23) | hadr_state - HADR State monitor element |
| HADR_FLAGS | VARCHAR(512) | hadr_flags - HADR flags monitor element |
| PRIMARY_MEMBER_HOST | VARCHAR(255) | primary_member_host - Primary member host monitor element |
| PRIMARY_INSTANCE | VARCHAR(128) | primary_instance - Primary instance monitor element |
| PRIMARY_MEMBER | SMALLINT | primary_member - Primary member monitor element |
| STANDBY_MEMBER_HOST | VARCHAR(255) | standby_member_host - Standby member hostname monitor element |
| STANDBY_INSTANCE | VARCHAR(128) | standby_instance - Standby instance monitor element |
| STANDBY_MEMBER | SMALLINT | standby_member - Standby member monitor element |
| HADR_CONNECT_STATUS | VARCHAR(12) | hadr_connect_status - HADR Connection Status monitor element |
| HADR_CONNECT_STATUS_TIME | TIMESTAMP | hadr_connect_status_time - HADR connect status time monitor element |
| HEARTBEAT_INTERVAL | BIGINT | heartbeat_interval - Heartbeat interval monitor element |
| HADR_TIMEOUT | BIGINT | hadr_timeout - HADR Timeout monitor element |
| TIME_SINCE_LAST_RECV | BIGINT | time_since_last_recv - Time since last message received monitor element |
| PEER_WAIT_LIMIT | BIGINT | peer_wait_limit - Peer wait limit monitor element |
| LOG_HADR_WAIT_CUR | BIGINT | log_hadr_wait_cur - Current logger wait time monitor element |
| LOG_HADR_WAIT_TIME | BIGINT | log_hadr_wait_time - Total logger wait time monitor element |
| LOG_HADR_WAITS_TOTAL | BIGINT | log_hadr_waits_total - Total logger wait events monitor element |
| SOCK_SEND_BUF_REQUESTED | BIGINT | sock_send_buf_requested - Requested socket send buffer size monitor element |
| SOCK_SEND_BUF_ACTUAL | BIGINT | sock_send_buf_actual - Actual socket send buffer size monitor element |

Table 133. Information returned for MON_GET_HADR (continued)

| Column Name | Data Type | Description |
|-----------------------------------|-------------|---|
| SOCK_RECV_BUF_REQUESTED | BIGINT | sock_recv_buf_requested - Requested socket receive buffer size monitor element |
| SOCK_RECV_BUF_ACTUAL | BIGINT | sock_recv_buf_actual - Actual socket receive buffer size monitor element |
| PRIMARY_LOG_FILE | VARCHAR(12) | primary_log_file - Primary log file monitor element |
| PRIMARY_LOG_PAGE | BIGINT | primary_log_page - Primary log page monitor element |
| PRIMARY_LOG_POS | BIGINT | primary_log_pos - Primary log position monitor element |
| PRIMARY_LOG_TIME | TIMESTAMP | primary_log_time - Primary log time monitor element |
| STANDBY_LOG_FILE | VARCHAR(12) | standby_log_file - Standby log file monitor element |
| STANDBY_LOG_PAGE | BIGINT | standby_log_page - Standby log page monitor element |
| STANDBY_LOG_POS | BIGINT | standby_log_pos - Standby log position monitor element |
| STANDBY_LOG_TIME | TIMESTAMP | standby_log_time - Standby log time monitor element |
| HADR_LOG_GAP | BIGINT | hadr_log_gap - HADR Log Gap monitor element |
| STANDBY_REPLAY_LOG_FILE | VARCHAR(12) | standby_replay_log_file - Standby replay log file monitor element |
| STANDBY_REPLAY_LOG_PAGE | BIGINT | standby_replay_log_page - Standby replay log page monitor element |
| STANDBY_REPLAY_LOG_POS | BIGINT | standby_replay_log_pos - Standby replay log position monitor element |
| STANDBY_REPLAY_LOG_TIME | TIMESTAMP | standby_replay_log_time - Standby replay log time monitor element |
| STANDBY_RECV_REPLAY_GAP | BIGINT | standby_recv_replay_gap - Standby receive replay gap monitor element |
| STANDBY_REPLAY_DELAY | BIGINT | standby_replay_delay - Standby replay delay monitor element |
| STANDBY_RECV_BUF_SIZE | BIGINT | standby_recv_buf_size - Standby receive buffer size monitor element |
| STANDBY_RECV_BUF_PERCENT | DOUBLE | standby_recv_buf_percent - Standby receive buffer percentage monitor element |
| STANDBY_SPOOL_LIMIT | BIGINT | standby_spool_limit - Standby spool limit monitor element |
| STANDBY_SPOOL_PERCENT | DOUBLE | standby_spool_percent - Standby spool percentage monitor element |
| PEER_WINDOW | BIGINT | peer_window - Peer window monitor element |
| PEER_WINDOW_END | TIMESTAMP | peer_window_end - Peer window end |
| TAKEOVER_APP_REMAINING_PRIMARY | BIGINT | takeover_app_remaining_primary - Applications remaining on primary monitor element |
| TAKEOVER_APP_REMAINING_STANDBY | BIGINT | takeover_app_remaining_standby - Applications remaining on standby monitor element |
| READS_ON_STANDBY_ENABLED | CHAR(1) | reads_on_standby_enabled - reads on standby enabled monitor element |
| STANDBY_REPLAY_ONLY_WINDOW_ACTIVE | CHAR(1) | standby_replay_only_window_active - Standby replay only window active monitor element |
| STANDBY_REPLAY_ONLY_WINDOW_START | TIMESTAMP | standby_replay_only_window_start - Replay only window start time monitor element |

Table 133. Information returned for MON_GET_HADR (continued)

| Column Name | Data Type | Description |
|---------------------------------------|-----------|---|
| STANDBY_REPLAY_ONLY_WINDOW_TRAN_COUNT | BIGINT | standby_replay_only_window_tran_count - Replay only window transactions monitor element |
| HEARTBEAT_MISSED | INTEGER | heartbeat_missed - Heartbeats missed monitor element |
| HEARTBEAT_EXPECTED | INTEGER | heartbeat_expected - Heartbeats expected monitor element |
| STANDBY_ERROR_TIME | TIMESTAMP | standby_error_time - Standby error time monitor element |

Usage notes

HADR pair view

Certain fields are applicable to primary or standby only. For example, PEER_WAIT_LIMIT is applicable only to primary, STANDBY_RECV_BUF_SIZE, STANDBY_SPOOL_LIMIT, READS_ON_STANDBY_ENABLED are applicable only to standby. When this kind of information is reported, the database currently in the role is used (which may be the remote database), rather than the local database. For example, PEER_WAIT_LIMIT seen on a standby database is the value configured on the primary database, not the local config of standby database (which will be used only when the standby turns into primary).

Information about remote database

Primary and standby exchange monitoring information via heartbeat messages. Therefore information about the remote database can be slightly out of date. See heartbeat interval (reported in table function) to estimate timeliness of information (network latency can add additional delay). If a database has never connected to its partner database since activation, information about remote database may be returned as SQL NULL to indicate "unknown".

Log shipping channel end points

The end points for a log shipping channel is uniquely identified by host, instance and member:

- Primary side: PRIMARY_MEMBER_HOST, PRIMARY_INSTANCE, PRIMARY_MEMBER
- Standby side: STANDBY_MEMBER_HOST, STANDBY_INSTANCE, STANDBY_MEMBER

Until a connection is made, end point information of remote end may not be available. When information not available, empty strings will be returned for host and instance names and zero returned for member ID. In addition when in a Db2 Enterprise Server Edition environment, 0 is always returned for the member ID.

Note on unit of time duration

Per monitor table function convention, all MON_GET_HADR time duration fields use milliseconds as unit. For those fields reflecting a configuration parameter (such as HADR_TIMEOUT, HADR_PEER_WINDOW) whose unit in configuration is seconds, the number returned by MON_GET_HADR table function will be different from the number used in **db2 get/update db cfg** command, and the number returned by SYSIBMADM.DBCFG admin view or SYSPROC.DB_GET_CFG() table function. For example, for a 60 second

HADR_TIMEOUT value, MON_GET_HADR will return 60000, while the configuration oriented interfaces will return 60. To convert the millisecond number to second, use column_name/1000 in your query.

Usage during takeover

During takeover, there may be a period when clients cannot connect to either primary or standby database. The recommended monitoring method during takeover is **db2pd -hadr**.

Column Order and Groups:

1. Cluster level summary: HADR_ROLE, REPLAY_TYPE, HADR_SYNCMODE.
2. Log stream level summary: STANDBY_ID, LOG_STREAM_ID, HADR_STATE
3. Log shipping channel end points:
 - a. Primary side: PRIMARY_MEMBER_HOST, PRIMARY_INSTANCE, PRIMARY_MEMBER
 - b. Standby side: STANDBY_MEMBER_HOST, STANDBY_INSTANCE, STANDBY_MEMBER

The end points uniquely identify an HADR log shipping channel in all scenarios. Host, instance or MEMBER_ID uniquely identifies a member.
4. Connection details:
 - a. Status: HADR_CONNECT_STATUS, HADR_CONNECT_STATUS_TIME
 - b. Network timing: HEARTBEAT_INTERVAL, HADR_TIMEOUT, TIME_SINCE_LAST_RECV
 - c. Logger wait timing: PEER_WAIT_LIMIT, LOG_HADR_WAIT_CUR, LOG_HADR_WAIT_TIME, LOG_HADR_WAITS_TOTAL
 - d. TCP buffer size: SOCK_SEND_BUF_REQUESTED, SOCK_SEND_BUF_ACTUAL, SOCK_RECV_BUF_REQUESTED, SOCK_RECV_BUF_ACTUAL
5. Log position details:
 - a. Primary log position: PRIMARY_LOG_FILE, PRIMARY_LOG_PAGE, PRIMARY_LOG_POS, PRIMARY_LOG_TIME
 - b. Standby log receive position: STANDBY_LOG_FILE, STANDBY_LOG_PAGE, STANDBY_LOG_POS, STANDBY_LOG_TIME
 - c. Primary-standby log gap: HADR_LOG_GAP
 - d. Standby log replay position: STANDBY_REPLAY_LOG_FILE, STANDBY_REPLAY_LOG_PAGE, STANDBY_REPLAY_LOG_POS, STANDBY_REPLAY_LOG_TIME
 - e. Standby receive-replay gap: STANDBY_RECV_REPLAY_GAP
 - f. Replay delay: STANDBY_REPLAY_DELAY
6. Log buffer and spooling: STANDBY_RECV_BUF_SIZE, STANDBY_RECV_BUF_PERCENT, STANDBY_SPOOL_LIMIT
7. Peer window: PEER_WINDOW, PEER_WINDOW_END
8. Takeover: TAKEOVER_APP_REMAINING_PRIMARY, TAKEOVER_APP_REMAINING_STANDBY
9. Reads on Standby: READS_ON_STANDBY_ENABLED, STANDBY_REPLAY_ONLY_WINDOW_ACTIVE, STANDBY_REPLAY_ONLY_WINDOW_START, STANDBY_REPLAY_ONLY_WINDOW_TRAN_COUNT

Examples

1.

```
SELECT HADR_ROLE, STANDBY_ID, HADR_STATE, varchar(PRIMARY_MEMBER_HOST ,20)
as PRIMARY_MEMBER_HOST, varchar(STANDBY_MEMBER_HOST ,20)
as STANDBY_MEMBER_HOST from table(MON_GET_HADR(NULL))
```

The following is an example of output from this query.

| HADR_ROLE | STANDBY_ID | HADR_STATE | PRIMARY_MEMBER_HOST |
|-----------|------------|----------------|---------------------|
| PRIMARY | 1 | PEER | hostP.ibm.com |
| PRIMARY | 2 | REMOTE_CATCHUP | hostP.ibm.com |
| PRIMARY | 3 | REMOTE_CATCHUP | hostP.ibm.com |

| STANDBY_MEMBER_HOST |
|---------------------|
| hostS1.ibm.com |
| hostS2.ibm.com |
| hostS3.ibm.com |

3 record(s) selected.

Query is issued to a primary database with 3 standbys in which 3 rows are returned. Each row represents a primary-standby log shipping channel. The HADR_ROLE column represents the role of the database to which the query is issued. Therefore it is PRIMARY on all rows.

2.

```
SELECT HADR_ROLE, STANDBY_ID, HADR_STATE, varchar(PRIMARY_MEMBER_HOST ,20)
as PRIMARY_MEMBER_HOST, varchar(STANDBY_MEMBER_HOST ,20)
as STANDBY_MEMBER_HOST from table(MON_GET_HADR(NULL))
```

The following is an example of output from this query.

| HADR_ROLE | STANDBY_ID | HADR_STATE | PRIMARY_MEMBER_HOST |
|-----------|------------|------------|---------------------|
| STANDBY | 0 | PEER | hostP.ibm.com |

| STANDBY_MEMBER_HOST |
|---------------------|
| hostS1.ibm.com |

1 record(s) selected.

Query is issued to a standby database with reads on standby enabled. Standby only knows about its own primary. Only one row is returned even if the standby is part of a multiple standby system. STANDBY_ID is always zero when query is issued to a standby.

3. The following example is for HADR in a Db2 pureScale environment only. The environment for this example has three member clusters. On the primary, member 0 is assisting member 1. Standby member 0 is the replay member.

- With member ID 0, the HADR information for log stream 0, and the assisted log stream (log stream 1) is output:

```
SELECT LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE, HADR_FLAGS
FROM TABLE (MON_GET_HADR(0))
```

| LOG_STREAM_ID | PRIMARY_MEMBER | STANDBY_MEMBER | HADR_STATE | HADR_FLAGS |
|---------------|----------------|----------------|----------------|-------------------------|
| 0 | 0 | 0 | PEER | |
| 1 | 0 | 0 | REMOTE_CATCHUP | ASSISTED_REMOTE_CATCHUP |

- With member ID 1, the HADR information for log stream 1 is output:

```
SELECT LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE, HADR_FLAGS
FROM TABLE (MON_GET_HADR(1))
```

| LOG_STREAM_ID | PRIMARY_MEMBER | STANDBY_MEMBER | HADR_STATE | HADR_FLAGS |
|---------------|----------------|----------------|--------------|------------|
| 1 | 1 | 0 | DISCONNECTED | |

- For log stream ID 2, the HADR information for log stream 2 is output:

```
SELECT LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE, HADR_FLAGS
FROM TABLE (MON_GET_HADR(2))
```

| LOG_STREAM_ID | PRIMARY_MEMBER | STANDBY_MEMBER | HADR_STATE | HADR_FLAGS |
|---------------|----------------|----------------|------------|------------|
| 2 | 2 | 0 | PEER | |

- When specifying -2 as member ID, HADR information on all log streams is reported. For assisted remote catchup, only the record from the assisting member is returned, and HADR_FLAGS indicates that it is in an assisted remote catchup state.

```
SELECT LOG_STREAM_ID, PRIMARY_MEMBER, STANDBY_MEMBER, HADR_STATE, HADR_FLAGS
FROM TABLE (MON_GET_HADR(-2))
```

| LOG_STREAM_ID | PRIMARY_MEMBER | STANDBY_MEMBER | HADR_STATE | HADR_FLAGS |
|---------------|----------------|----------------|----------------|-------------------------|
| 0 | 0 | 0 | PEER | |
| 1 | 0 | 0 | REMOTE_CATCHUP | ASSISTED_REMOTE_CATCHUP |
| 2 | 2 | 0 | PEER | |

MON_GET_INDEX table function - get index metrics

The MON_GET_INDEX table function returns metrics for one or more indexes.

Syntax

```
►► MON_GET_INDEX (—tabschema—, —tabname—, —member—) ►►
```

The schema is SYSPROC.

Table function parameters

tabschema

An input argument of type VARCHAR(128) that specifies a valid table schema name in the same database as the one currently connected to when calling this function. If the argument is NULL or an empty string, metrics are retrieved for indexes of tables in all schemas in the database. If the argument is specified, metrics are only returned for indexes for tables in the specified schema.

tabname

An input argument of type VARCHAR(128) that specifies a valid table name in the same database as the one currently connected to when calling this function. Metrics are returned for all indexes on the specified table. If the argument is null or an empty string, metrics are retrieved for all indexes for all tables in the database.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify a -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is set implicitly

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

Identify the most frequently used indexes on the DMEXT002.TABLE1 table, since the last database activation:

```
SELECT VARCHAR(S.INDSCHEMA, 10) AS INDSCHEMA,  
       VARCHAR(S.INDNAME, 10) AS INDNAME,  
       T.DATA_PARTITION_ID,  
       T.MEMBER,  
       T.INDEX_SCANS,  
       T.INDEX_ONLY_SCANS  
FROM TABLE(MON_GET_INDEX('DMEXT002','TABLE1', -2)) as T, SYSCAT.INDEXES AS S  
WHERE T.TABSCHEMA = S.TABSCHEMA AND  
      T.TABNAME = S.TABNAME AND  
      T.IID = S.IID  
ORDER BY INDEX_SCANS DESC
```

The following is an example of output from this query.

| INDSCHEMA | INDNAME | DATA_PARTITION_ID | MEMBER | INDEX_SCANS | INDEX_ONLY_SCANS |
|-----------|---------|-------------------|--------|-------------|------------------|
| DMEXT002 | INDEX3 | - | - | 0 | 1 |
| DMEXT002 | INDEX4 | - | - | 0 | 1 |
| DMEXT002 | INDEX1 | - | - | 0 | 0 |
| DMEXT002 | INDEX2 | - | - | 0 | 0 |
| DMEXT002 | INDEX5 | - | - | 0 | 0 |
| DMEXT002 | INDEX6 | - | - | 0 | 0 |

6 record(s) selected.

Usage notes

The MON_GET_INDEX table function returns one row of data per index, and per database member. If partitioned indexes are being used, one row is returned for each index partition per database member. No aggregation across database members is performed. However, aggregation can be achieved through SQL queries as shown in the previous example.

Metrics will only be returned for indexes on tables that have been accessed since the database was activated. All counters represent data since the current database activation. For example, the *pseudo_empty_pages* counter is the number of pages that have been identified as pseudo empty since the database was activated. It is not the current number of pseudo empty pages in the index.

The metrics returned by this function are collected at two levels. Some metrics are always collected, while the collection of other metrics is controlled at the database level using the *mon_obj_metrics* configuration parameter. Refer to the descriptions provided in the following table to determine if any settings must be active for data to be collected for a particular metric.

Information returned

Table 134. Information returned for MON_GET_INDEX

| Column Name | Data Type | Description or corresponding monitor element |
|-------------|--------------|--|
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TABNAME | VARCHAR(128) | table_name - Table name |
| IID | SMALLINT | iid - Index identifier |

Table 134. Information returned for MON_GET_INDEX (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|--|
| MEMBER | SMALLINT | member - Database member |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data partition identifier |
| NLEAF | BIGINT | nleaf - Number of leaf pages |
| NLEVELS | SMALLINT | nlevels - Number of index levels |
| INDEX_SCANS | BIGINT | index_scans - Index scans |
| INDEX_ONLY_SCANS | BIGINT | index_only_scans - Index-only scans |
| KEY_UPDATES | BIGINT | key_updates - Key updates |
| INCLUDE_COL_UPDATES | BIGINT | include_col_updates - Include column updates |
| PSEUDO_DELETES | BIGINT | pseudo_deletes - Pseudo deletes |
| DEL_KEYS_CLEANED | BIGINT | del_keys_cleaned - Pseudo deleted keys cleaned |
| ROOT_NODE_SPLITS | BIGINT | root_node_splits - Root node splits |
| INT_NODE_SPLITS | BIGINT | int_node_splits - Intermediate node splits |
| BOUNDARY_LEAF_NODE_SPLITS | BIGINT | boundary_leaf_node_splits - Boundary leaf node splits |
| NONBOUNDARY_LEAF_NODE_SPLITS | BIGINT | nonboundary_leaf_node_splits - Non-boundary leaf node splits |
| PAGE_ALLOCATIONS | BIGINT | page_allocations - Page allocations |
| PSEUDO_EMPTY_PAGES | BIGINT | pseudo_empty_pages - Pseudo empty pages |
| EMPTY_PAGES_REUSED | BIGINT | empty_pages_reused - Empty pages reused |
| EMPTY_PAGES_DELETED | BIGINT | empty_pages_deleted - Empty pages deleted |
| PAGES_MERGED | BIGINT | pages_merged - Pages merged |
| OBJECT_INDEX_L_READS | BIGINT | object_index_l_reads - Buffer pool index logical reads for an index |
| OBJECT_INDEX_P_READS | BIGINT | object_index_p_reads - Buffer pool index physical reads for an index |
| OBJECT_INDEX_GBP_L_READS | BIGINT | object_index_gbp_l_reads - Group buffer pool index logical reads for an index |
| OBJECT_INDEX_GBP_P_READS | BIGINT | object_index_gbp_p_reads - Group buffer pool index physical reads for an index |
| OBJECT_INDEX_GBP_INVALID_PAGES | BIGINT | object_index_gbp_invalid_pages - Group buffer pool invalid index pages for an index |
| OBJECT_INDEX_LBP_PAGES_FOUND | BIGINT | object_index_lbp_pages_found - Local buffer pool index pages found for an index |
| OBJECT_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | object_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool |
| INDEX_JUMP_SCANS | BIGINT | index_jump_scans - Index jump scans |
| OBJECT_INDEX_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| OBJECT_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| OBJECT_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |

Table 134. Information returned for MON_GET_INDEX (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|--|
| OBJECT_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |

MON_GET_INDEX_USAGE_LIST table function - Returns information from an index usage list

The MON_GET_INDEX_USAGE_LIST table function returns information from a usage list defined for an index.

Syntax

►►MON_GET_INDEX_USAGE_LIST(—*usagelistschema*—,—*usagelistname*—,—*member*—)◄◄

The schema is SYSPROC.

Table function parameters

usagelistschema

An input argument of type VARCHAR(128) that specifies a valid schema name in the currently connected database when calling this function. If the argument is null or an empty string, usage lists are retrieved in all schemas in the database. If the argument is specified, usage lists are only returned for the specified schema. The default is NULL.

usagelistname

An input argument of type VARCHAR(128) that specifies a usage list defined for an index that resides in the currently connected database when calling this function. If *usagelistname* is null or an empty string, then all usage lists defined for an index from the schemas identified by the *usagelistschema* that exist are retrieved. If specified, only the usage list specified from the schemas identified by the *usagelistschema* is returned. The default is NULL.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

Return the usage list USL_MON_IND1 from all database members

| USAGELISTSHEMA | USAGELISTNAME | INDSCHEMA | INDNAME | MEMBER |
|----------------|---------------|-----------|---------|--------|
| ISAYYID | USL_MON_IND1 | ISAYYID | I1 | 0 |
| ISAYYID | USL_MON_IND1 | ISAYYID | I1 | 0 |

```
EXECUTABLE_ID
-----
x'010000000000000049000000000000000000000020020110706093802577065'
x'01000000000000004B000000000000000000000020020110706093825981548'
```

| MON_INTERVAL_ID | LAST_UPDATED | NUM_REFERENCES |
|-----------------|-----------------------------|----------------|
| | 02011-07-06-09.38.15.881668 | 1 |
| | 02011-07-06-09.38.25.984147 | 1 |

| NUM_REF_WITH_METRICS | OBJECT_INDEX_L_READS | OBJECT_INDEX_P_READS |
|----------------------|----------------------|----------------------|
| 1 | 1 | 0 |
| 1 | 1 | 0 |

| OBJECT_INDEX_GBP_L_READS | OBJECT_INDEX_GBP_P_READS |
|--------------------------|--------------------------|
| 0 | 0 |
| 0 | 0 |

| OBJECT_INDEX_GBP_INVALID_PAGES | OBJECT_INDEX_LBP_PAGES_FOUND |
|--------------------------------|------------------------------|
| 0 | 0 |
| 0 | 0 |

Usage notes

Use the `num_ref_with_metrics` column instead of the `num_references` column when computing averages, since the `num_references` column counts all executions of the section, regardless of whether or not the execution of the section contributed to the metrics that are reported.

Metrics collected by this function are controlled at the database level using the `mon_obj_metrics` configuration parameter. By default, metrics collection is enabled.

Information returned

Table 135. Information returned for MON_GET_INDEX_USAGE_LIST

| Column Name | Data Type | Description |
|---|---------------------------|--|
| USAGELISTSHEMA | VARCHAR (128) | usage_list_schema - Usage list schema |
| USAGELISTNAME | VARCHAR (128) | usage_list_name - Usage list name |
| INDSCHEMA | VARCHAR (128) | index_schema - Index schema |
| INDNAME | VARCHAR (128) | index_name - Index name |
| MEMBER | SMALLINT | member - Database member |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data partition identifier |
| EXECUTABLE_ID | VARCHAR (32) FOR BIT DATA | executable_id - Executable ID |
| MON_INTERVAL_ID | BIGINT | mon_interval_id - Monitor interval identifier |
| LAST_UPDATED | TIMESTAMP | last_updated - Last update time stamp |
| NUM_REFERENCES | BIGINT | num_references - Number of references |
| NUM_REF_WITH_METRICS | BIGINT | num_ref_with_metrics - Number of references with metrics |
| OBJECT_INDEX_L_READS | BIGINT | object_index_l_reads - Buffer pool index logical reads for an index |
| OBJECT_INDEX_P_READS | BIGINT | object_index_p_reads - Buffer pool index physical reads for an index |
| OBJECT_INDEX_GBP_L_READS | BIGINT | object_index_gbp_l_reads - Group buffer pool index logical reads for an index |
| OBJECT_INDEX_GBP_P_READS | BIGINT | object_index_gbp_p_reads - Group buffer pool index physical reads for an index |
| OBJECT_INDEX_GBP_INVALID_PAGES | BIGINT | object_index_gbp_invalid_pages - Group buffer pool invalid index pages for an index |
| OBJECT_INDEX_LBP_PAGES_FOUND | BIGINT | object_index_lbp_pages_found - Local buffer pool index pages found for an index |
| OBJECT_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | object_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool |
| OBJECT_INDEX_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| OBJECT_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| OBJECT_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| OBJECT_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |

MON_GET_INSTANCE table function - Get instance level information

The MON_GET_INSTANCE table function returns information about the database instance level.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_GET_INSTANCE—(—*member*—)————►►

The schema is SYSPROC.

Routine parameters

member

An input parameter of type INTEGER that specifies a valid database member from which the records should be fetched. Specify -1 or null for the current member, or -2 for information from all active database members. An active database member is a database that is available for connection and use by applications.

Information returned

Table 136. Information returned by the MON_GET_INSTANCE routine

| Column name | Data type | Description |
|-------------------|--------------|--|
| MEMBER | SMALLINT | member - Database member monitor element |
| DB2_STATUS | VARCHAR(12) | db2_status - Status of Db2 instance monitor element |
| DB2START_TIME | TIMESTAMP | db2start_time - Start Database Manager Timestamp monitor element |
| TIMEZONEOFFSET | INTEGER | timezoneoffset - Time difference from UCT monitor element |
| TIMEZONEID | VARCHAR(255) | timezoneid - Time zone identifier monitor element |
| CON_LOCAL_DBASES | BIGINT | con_local_dbases - Local Databases with Current Connects monitor element |
| TOTAL_CONNECTIONS | BIGINT | total_connections - Total connections |
| AGENTS_REGISTERED | BIGINT | agents_registered - Agents Registered monitor element |

Table 136. Information returned by the MON_GET_INSTANCE routine (continued)

| Column name | Data type | Description |
|---------------------------|--------------|--|
| AGENTS_REGISTERED_TOP | BIGINT | agents_registered_top - Maximum Number of Agents Registered monitor element |
| IDLE_AGENTS | BIGINT | idle_agents - Number of Idle Agents monitor element |
| AGENTS_FROM_POOL | BIGINT | agents_from_pool - Agents Assigned From Pool monitor element |
| AGENTS_CREATED_EMPTY_POOL | BIGINT | agents_created_empty_pool - Agents Created Due to Empty Agent Pool monitor element |
| NUM_COORD_AGENTS | BIGINT | num_coord_agents - Number of coordinator agents |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum Number of Coordinating Agents monitor element |
| AGENTS_STOLEN | BIGINT | agents_stolen - Stolen Agents monitor element |
| GW_TOTAL_CONS | BIGINT | gw_total_cons - Total Number of Attempted Connections for Db2 Connect monitor element |
| GW_CUR_CONS | BIGINT | gw_cur_cons - Current Number of Connections for Db2 Connect monitor element |
| GW_CONS_WAIT_HOST | BIGINT | gw_cons_wait_host - Number of Connections Waiting for the Host to Reply monitor element |
| GW_CONS_WAIT_CLIENT | BIGINT | gw_cons_wait_client - Number of Connections Waiting for the Client to Send Request monitor element |
| NUM_GW_CONN_SWITCHES | BIGINT | num_gw_conn_switches - Connection Switches monitor element |
| PRODUCT_NAME | VARCHAR(32) | product_name - Product Name monitor element |
| SERVICE_LEVEL | VARCHAR(64) | service_level - Service Level monitor element |
| SERVER_PLATFORM | VARCHAR(32) | server_platform - Server Operating System monitor element |
| NETWORK_INTERFACE_BOUND | VARCHAR(255) | network_interface_bound - Hostname of member representing a network interface for communications |

Example

Determine the start time and total number of current connections on all members:

```
SELECT DB2START_TIME, TOTAL_CONNECTIONS
FROM TABLE (MON_GET_INSTANCE(-2));
```

This query returns the following output:

| DB2START_TIME | TOTAL_CONNECTIONS |
|----------------------------|-------------------|
| 2012-09-11-15.39.05.000000 | 2 |

MON_GET_LATCH table function - List all latches in the current member

The MON_GET_LATCH table function returns a list of all latches in the current member.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

►► MON_GET_LATCH(—*search_args*—,—*member*—)◄◄

The schema is SYSPROC.

Routine parameters

search_args

An input parameter of type CLOB(1 K) that represents a list of *key-value* pairs. If the list is empty or NULL, all latches from the specified member are returned. Otherwise, all latches that match all of the conditions that are represented by the list of *key-value* pairs are returned. A *key-value* pair must follow this format:

- A *key* is a string that represents an opening tag. An opening tag consists of an angle bracket, followed by the key name, followed by a closing angle bracket. Spaces are not allowed. An example of an opening tag is <application_handle>
- The *value* is placed after the opening tag
- A closing tag consists of an opening angle bracket, followed by a forward slash, followed by the key name, followed by a closing angle bracket. Spaces are not allowed. An example of a closing tag is </application_handle>

The following example shows of a full *key-value* pair:

```
'<application_handle>123</application_handle>'
```

All keys are case-sensitive and can be specified one time each in the **search_args** parameter.

The order of the keys does not matter.

SQLCODE -171 is returned for an invalid *key-value* pair.

An implicit AND operator exists between different keys.

The following use of the **search_args** parameter returns a list of all latches that are held or waiting to be acquired by the application with the handle 123:

```
CLOB('<application_handle>123</application_handle>')
```

The following keys are available for the **search_args** parameter:

application_handle

Returns a list of all latches that are currently held, or are being acquired by, the specified application handle. The value is specified as an INTEGER. For example:

```
CLOB('<application_handle>145</application_handle>')
```

latch_name

Returns a list of all latches that match the specified latch name. The value is specified as a string, whose maximum length is 256 characters. For example:

```
CLOB('<latch_name>SQLO_LT_SQLB_PTBL__pool_table_latch</latch_name>')
```

edu_id

Returns a list of all latches that match the specified *edu_id*. For example:

```
CLOB('<edu_id>28</edu_id>')
```

latch_status

Returns a list of all latches in the specified status. The value that is specified can be one of the following characters:

H Latches that are held.

W Latches that are waiting.

C Latches that are contested. A contested latch occurs when an application acquires a latch, then another application waits for this latch.

For example:

```
CLOB('<latch_status>W</latch_status>')
```

The following examples demonstrate how to use *key-value* pairs in the **search_args** parameter.

1. Search for all latches:

```
CLOB('')
```

2. Search for a latch that is named *SQLQ_LT_SQLB_PTBL__pool_table_latch*, and an application with a handle of 123 is holding or waiting for that latch:

```
CLOB('<application_handle>123</application_handle>  
<latch_name>SQLQ_LT_SQLB_PTBL__pool_table_latch</latch_name>')
```

3. Search for all latches that are contested:

```
CLOB('<latch_status>C</latch_status>')
```

member

An input argument of type INTEGER that specifies the member from which data is returned. Specify -1 for the current member. Specify -2 for all active members.

Information returned

Table 137. Information returned by the MON_GET_LATCH table function

| Column name | Data type | Description |
|--------------------|--------------|---|
| LATCH_NAME | VARCHAR(256) | latch_name - Latch name monitor element |
| MEMORY_ADDRESS | VARCHAR(32) | memory_address - Memory address monitor element |
| EDU_ID | BIGINT | edu_id - Engine dispatchable unit ID monitor element |
| EDU_NAME | VARCHAR(128) | edu_name - Engine dispatchable unit name monitor element |
| APPLICATION_HANDLE | BIGINT | <p>application_handle - Application handle monitor element</p> <p>If the value in the LATCH_STATUS column is 'H', APPLICATION_HANDLE represents the application that is holding the latch.</p> <p>If the value in the LATCH_STATUS column is 'W', APPLICATION_HANDLE represents the application that is waiting on the latch.</p> |
| MEMBER | SMALLINT | member - Database member monitor element |
| LATCH_STATUS | CHAR(1) | latch_status - Latch status monitor element |
| LATCH_WAIT_TIME | BIGINT | latch_wait_time - Latch wait time monitor element |

Example

In this sample scenario, the MON_GET_LATCH table functions are used to get all latch information in databases with connections, on all members.

1. Call the MON_GET_LATCH table function to retrieve all latch information in databases with connections, on all members:

```
SELECT SUBSTR(LATCH_NAME,1,40) LATCH_NAME,
       SUBSTR(MEMORY_ADDRESS,1,20) MEMORY_ADDRESS,
       EDU_ID,
       SUBSTR(EDU_NAME,1,20) EDU_NAME,
       APPLICATION_HANDLE,
       MEMBER,
```

```

        LATCH_STATUS,
        LATCH_WAIT_TIME
FROM TABLE ( MON_GET_LATCH( NULL, -2 ) ) ORDER BY LATCH_NAME, LATCH_STATUS

```

This query returns the following output:

| LATCH_NAME | MEMORY_ADDRESS | EDU_ID | ... |
|---------------------------------------|--------------------|--------|-----|
| SQLQ_LT_SQLB_POOL_CB_ptfLotch | 0x70000005351A440 | 36114 | ... |
| SQLQ_LT_SQLB_POOL_CB_readLotch | 0x70000005351A3C0 | 36114 | ... |
| SQLQ_LT_SQLB_POOL_CB_readLotch | 0x70000005351A3C0 | 37911 | ... |
| SQLQ_LT_SQLB_PTBL_pool_table_latch | 0x70000004108B910 | 37911 | ... |
| SQLQ_LT_SQLB_PTBL_pool_table_latch | 0x70000004108B910 | 37654 | ... |
| SQLQ_LT_SQLB_PTBL_pool_table_latch | 0x70000004108B910 | 37140 | ... |
| SQLQ_LT_SQLE_KRCB_EDUChainLatch | 0x780000000472EAC | 37397 | ... |
| SQLQ_LT_preventSuspendIOLotch | 0x78000000046081C0 | 36114 | ... |
| SQLQ_LT_sqeWLDDispatcher_m_tunerLatch | 0x7800000001C68440 | 1029 | ... |

Output for query (continued):

| ... EDU_NAME | APPLICATION_HANDLE | MEMBER | LATCH_STATUS | LATCH_WAIT_TIME |
|-----------------------|--------------------|--------|--------------|-----------------|
| ... | | | | |
| ... db2agent (SAMPLE) | 118 | 0 H | | - |
| ... db2agent (SAMPLE) | 118 | 0 H | | - |
| ... db2agent (SAMPLE) | 124 | 0 W | | 65850 |
| ... db2agent (SAMPLE) | 124 | 0 H | | - |
| ... db2agent (SAMPLE) | 121 | 0 W | | 50397 |
| ... db2agent (SAMPLE) | 119 | 0 W | | 30886 |
| ... db2agent (SAMPLE) | 120 | 0 H | | - |
| ... db2agent (SAMPLE) | 118 | 0 H | | - |
| ... db2w1mt | - | 0 H | | - |

9 record(s) selected.

2. Call the MON_GET_LATCH table function to determine the latches that are contested.

```

SELECT SUBSTR(LATCH_NAME,1,40) LATCH_NAME,
       SUBSTR(MEMORY_ADDRESS,1,20) ADDRESS,
       EDU_ID,
       SUBSTR(EDU_NAME,1,20) EDU_NAME,
       APPLICATION_HANDLE,
       MEMBER,
       LATCH_STATUS,
       LATCH_WAIT_TIME
FROM TABLE ( MON_GET_LATCH( CLOB('<LATCH_STATUS>C</LATCH_STATUS>'), -2 ) )
ORDER BY LATCH_NAME, LATCH_STATUS

```

This query returns the following output:

| LATCH_NAME | ADDRESS | EDU_ID | ... |
|------------------------------------|-------------------|--------|-----|
| SQLQ_LT_SQLB_POOL_CB_readLotch | 0x70000005351A3C0 | 36114 | ... |
| SQLQ_LT_SQLB_POOL_CB_readLotch | 0x70000005351A3C0 | 37911 | ... |
| SQLQ_LT_SQLB_PTBL_pool_table_latch | 0x70000004108B910 | 37911 | ... |
| SQLQ_LT_SQLB_PTBL_pool_table_latch | 0x70000004108B910 | 37654 | ... |
| SQLQ_LT_SQLB_PTBL_pool_table_latch | 0x70000004108B910 | 37140 | ... |

Output for query (continued):

| ... EDU_NAME | APPLICATION_HANDLE | MEMBER | LATCH_STATUS | LATCH_WAIT_TIME |
|-----------------------|--------------------|--------|--------------|-----------------|
| ... | | | | |
| ... db2agent (SAMPLE) | 118 | 0 H | | - |
| ... db2agent (SAMPLE) | 124 | 0 W | | 78140 |
| ... db2agent (SAMPLE) | 124 | 0 H | | - |
| ... db2agent (SAMPLE) | 121 | 0 W | | 62686 |
| ... db2agent (SAMPLE) | 119 | 0 W | | 43175 |

5 record(s) selected.

- The previous output shows two latches that are contested by applications. To retrieve only the `SQLQ_LT_SQLB_PTBL__pool_table_latch` latch, specify the **latch_name** value in the **search_args** argument to return the applications that are contesting this latch.

```
SELECT SUBSTR(LATCH_NAME,1,40) LATCH_NAME,
       SUBSTR(MEMORY_ADDRESS,1,20) ADDRESS,
       EDU_ID,
       SUBSTR(EDU_NAME,1,20) EDU_NAME,
       APPLICATION_HANDLE,
       MEMBER,
       LATCH_STATUS,
       LATCH_WAIT_TIME
FROM TABLE (
  MON_GET_LATCH( CLOB(' <LATCH_STATUS>C</LATCH_STATUS>
                    <LATCH_NAME>SQLQ_LT_SQLB_PTBL__POOL_TABLE_LATCH</LATCH_NAME>' ), -2 )
)
ORDER BY LATCH_NAME, LATCH_STATUS
```

This query returns the following output:

| LATCH_NAME | ADDRESS | EDU_ID | ... |
|-------------------------------------|-------------------|--------|-----|
| SQLQ_LT_SQLB_PTBL__pool_table_latch | 0x70000004108B910 | 37911 | ... |
| SQLQ_LT_SQLB_PTBL__pool_table_latch | 0x70000004108B910 | 37654 | ... |
| SQLQ_LT_SQLB_PTBL__pool_table_latch | 0x70000004108B910 | 37140 | ... |

Output for query (continued):

| ... EDU_NAME | APPLICATION_HANDLE | MEMBER | LATCH_STATUS | LATCH_WAIT_TIME |
|-----------------------|--------------------|--------|--------------|-----------------|
| ... db2agent (SAMPLE) | 124 | 0 | H | - |
| ... db2agent (SAMPLE) | 121 | 0 | W | 74956 |
| ... db2agent (SAMPLE) | 119 | 0 | W | 55446 |

3 record(s) selected.

MON_GET_LOCKS - List all locks in the currently connected database

The `MON_GET_LOCKS` table function returns a list of all locks in the currently connected database.

To get information about locks, use the `MON_GET_LOCKS`, `MON_FORMAT_LOCK_NAME`, and `MON_GET_APPL_LOCKWAIT` table functions, and the `MON_LOCKWAIT` administrative view instead of the `SNAPLOCKWAIT` administrative view and `SNAP_GET_LOCKWAIT` table function, the `SNAPLOCK` administrative view and `SNAP_GET_LOCK` table function, and the `LOCKS_HELD` administrative view which are deprecated in Fix Pack 1 of Version 9.7.

►►—MON_GET_LOCKS—(—search_args—, —member—)—————◄◄

The schema is `SYSPROC`.

Table function parameters

search_args

An input parameter of type `CLOB(1K)` that represents a list of *key-value* pairs. If the list is empty or `NULL`, all locks in the currently connected database are returned. Otherwise, all locks that match all of the conditions represented by the list of *key-value* pairs are returned. A *key-value* pair must follow this format:

- A *key* is a string that consists of an opening tag, followed by the value, followed by a closing tag.
- An opening tag consists of an opening angle bracket, followed by the key name, followed by a closing angle bracket. No spaces are allowed.
- A closing tag consists of an opening angle bracket, followed by a forward slash, followed by the key name, followed by a closing angle bracket. No spaces are allowed.
- All keys are case-sensitive and can only be specified once in the *search_args* parameter.
- The order of the keys does not matter.

SQLCODE -171 is returned for an invalid *key-value* pair.

SQLCODE -204 is returned if the table does not exist.

An AND operation is performed between different keys. An OR operation is performed between multiple values of the same key. For example, the following use of the *search_args* parameter returns a list of all locks of type Table or Row, that are held, or waiting to be acquired, in either Shared or Exclusive mode, by the application with the handle 123:

```
CLOB('<application_handle>123</application_handle>
      <lock_object_type>Table:Row</lock_object_type>
      <lock_mode>S:X</lock_mode>')
```

The available keys for the MON_GET_LOCKS table function are as follows:

- **application_handle**
Returns a list of all locks that are currently held or are in the process of being acquired by the specified application handle. Only a single occurrence of the key value can be specified. The value is specified as an INTEGER. For example:
CLOB('<application_handle>145</application_handle>')
- **lock_name**
Returns a list of all locks that match the specified lock name. Only a single occurrence of the key value can be specified. The value is specified as a string of maximum length 32. For example:
CLOB('<lock_name>00030005000000000280000452</lock_name>')
- **lock_object_type**
Returns a list of all locks that match the specified lock object type. Multiple occurrences of the key value can be specified (to a maximum of 5). Each value (case insensitive) must be separated by a colon (:) and is specified as a string of a maximum length of 32 characters. For example:
CLOB('<lock_object_type>Table:Chunk:Plan</lock_object_type>')
For a list of possible input values, see “lock_object_type - Lock object type waited on monitor element”.
- **lock_mode**
Returns a list of all locks that match the specified lock mode. Multiple occurrences of the key value can be specified (to a maximum of 5). Each value (case insensitive) is separated by a colon (:) and is specified as a string of maximum length 3. For example:
CLOB('<lock_mode>IS:IN:U</lock_mode>')
For a list of possible input values, see “lock_mode - Lock mode monitor element”.
- **lock_status**

Returns a list of all locks in the specified status. Only a single occurrence of the key value can be specified. The value is specified as a character.

```
CLOB('<lock_status>W</lock_status>')
```

For a list of possible input values, see “lock_status - Lock status monitor element”.

- **table_schema**

Returns a list of all locks that are qualified by the specified schema name. The table_name key must also be specified. Only a single occurrence of the key value can be specified. The value is specified as a string of maximum length 128.

- **table_name**

Returns a list of all locks that reference the specified table. The table_schema key must also be specified. Only a single occurrence of the key value can be specified. The value is specified as a string of maximum length 128. For example:

```
CLOB('<table_schema>USER1</table_schema>
<table_name>INVENTORY</table_name>')
```

The following examples demonstrate how to use *key-value* pairs in the *search_args* parameter.

1. To search for all ROW and TABLE locks:

```
CLOB('<lock_object_type>Table:Row</lock_object_type>')
```

2. To search for all locks that application handle 123 is holding or waiting to acquire that reference table T1, and were created by user USER1:

```
CLOB('<application_handle>123</application_handle>
<table_schema>USER1</table_schema>
<table_name>T1</table_name>')
```

3. To search for all TABLE, ROW, and BUFFERPOOL locks that are currently held in Shared mode:

```
CLOB('<lock_mode>S</lock_mode>
<lock_status>G</lock_status>
<lock_object_type>Table:Row:Bufferpool</lock_object_type>')
```

member

An input argument of type INTEGER that specifies from which member the data is returned. Specify -1 for the current member, and -2 for all active members.

Authorization

One of the following authorities or privilege is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

In this sample scenario, the MON_GET_LOCKS and MON_GET_APPL_LOCKWAIT table functions are used to investigate the locking situation in the current connected database, on all members.

1. Call the MON_GET_APPL_LOCKWAIT table function to determine all the locks that are waiting to be acquired in the current connected database, on all members:

```
SELECT lock_name,  
       hld_member,  
       lock_status,  
       hld_application_handle FROM  
TABLE (MON_GET_APPL_LOCKWAIT(NULL, -2))
```

This query returns the following output:

| LOCK_NAME | HLD_MEMBER | LOCK_STATUS | HLD_APPLICATION_HANDLE |
|----------------------------|------------|-------------|------------------------|
| 00030005000000000280000452 | -2 | W | |
| 00030005000000000280000452 | -2 | W | |
| 00030005000000000280000452 | -2 | W | |

3 record(s) selected.

The records that show HLD_MEMBER is -2 indicate that the lock 0x00030005000000000280000452 is being held at a remote member.

2. Call the MON_GET_LOCKS table function to determine the holder of the lock, by specifying the lock name, 0x00030005000000000280000452, as the search argument:

```
SELECT lock_name,  
       member,  
       lock_status,  
       application_handle FROM  
TABLE (MON_GET_LOCKS(  
CLOB('<lock_name>00030005000000000280000452</lock_name>'),  
-2))
```

This query returns the following output:

| LOCK_NAME | MEMBER | LOCK_STATUS | APPLICATION_HANDLE |
|----------------------------|--------|-------------|--------------------|
| 00030005000000000280000452 | 0 | W | 12562 |
| 00030005000000000280000452 | 1 | W | 12562 |
| 00030005000000000280000452 | 2 | G | 65545 |
| 00030005000000000280000452 | 3 | W | 12562 |

4 record(s) selected.

To find out more about the application holding the lock, you can call the WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES or WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table functions.

Information returned

Table 138. Information returned by the MON_GET_LOCKS table function

| Column name | Data type | Description or monitor element |
|---------------------|----------------------|---|
| APPLICATION_HANDLE | BIGINT | <p>application_handle - Application handle</p> <p>If the LOCK_STATUS column is G, this represents the application that is currently holding the lock.</p> <p>If the LOCK_STATUS column is W or C, this represents the application that is currently waiting to acquire the lock.</p> <p>The value 0 indicates that the application no longer exists. The value can appear when incomplete transactions are on the database after a system crash. It means that the lock is held by either a transaction that is being rolled back or by an indoubt transaction that must be resolved.</p> |
| MEMBER | SMALLINT | member - Database member from which the data was retrieved for this row. |
| LOCK_NAME | VARCHAR(32) | lock_name - Lock name |
| LOCK_OBJECT_TYPE_ID | CHAR(1) FOR BIT DATA | Reserved for future use |
| LOCK_OBJECT_TYPE | VARCHAR(32) | <p>lock_object_type - Lock object type</p> <p>If the LOCK_STATUS column is G, this represents the type of object that the application is currently holding.</p> <p>If the LOCK_STATUS column is W or C, then this represents the type of object that the application is currently waiting to acquire.</p> <p>For possible input values, see “lock_object_type - Lock object type waited on monitor element”.</p> |

Table 138. Information returned by the MON_GET_LOCKS table function (continued)

| Column name | Data type | Description or monitor element |
|--------------------|------------|---|
| LOCK_MODE | VARCHAR(3) | lock_mode - Lock mode

If the LOCK_STATUS column is G, this represents the mode that the application is currently holding the lock in.

If the LOCK_STATUS column is W or C, this represents the mode that the application is currently waiting to acquire the lock in.

If the mode is unknown, a value of NULL is returned for this column. |
| LOCK_CURRENT_MODE | VARCHAR(3) | lock_current_mode - Original Lock Mode Before Conversion

If the mode is unknown, a value of NULL is returned for this column. |
| LOCK_STATUS | CHAR(1) | lock_status - Lock status |
| LOCK_ATTRIBUTES | CHAR(16) | lock_attributes - Lock attributes |
| LOCK_RELEASE_FLAGS | CHAR(16) | lock_release_flags - Lock release flags monitor element |
| LOCK_RRIID | BIGINT | Reserved for internal use
Reserved for internal use |
| LOCK_COUNT | BIGINT | Lock_count monitor element |
| LOCK_HOLD_COUNT | BIGINT | lock_hold_count monitor element |
| TBSP_ID | BIGINT | tablespace_id - Table space ID

For locks that do not reference a table space, a value of NULL is returned. |
| TAB_FILE_ID | BIGINT | table_file_id - Table file ID |

MON_GET_MEMORY_POOL - get memory pool information

The MON_GET_MEMORY_POOL table function retrieves metrics from the memory pools contained within a memory set.

Syntax

►► MON_GET_MEMORY_POOL (—memory_set_type—, —db_name—, —member—) ◀◀

The schema is SYSPROC.

Table function parameters

memory_set_type

An input argument of type VARCHAR(32) that specifies the type of the memory set when calling this function. If the argument is NULL or an empty string, then metrics are retrieved for all memory sets at the instance and database level. Otherwise metrics for the specified memory set are retrieved.

These parameter values are accepted:

| Value | Scope | Description |
|-------------|--|--|
| DBMS | Instance | Database manager (DBM) memory set |
| FMP | Instance | Fenced mode process (FMP) memory set |
| PRIVATE | Instance | Private memory set |
| DATABASE | Database | Database memory set |
| APPLICATION | Database | Application memory set |
| FCM | Host - there is only one FCM memory set allocated per machine for an instance. | Fast communication manager (FCM) memory set |
| NULL | All | All memory sets at the instance and database level |

db_name

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database when calling this function.

The database must have a directory entry type of either “INDIRECT” or “HOME”, as returned by a LIST DATABASE DIRECTORY command. The database must be active. Alternatively, the CURRENT_SERVER special register can be specified to retrieve metrics from the currently connected database. The register value contains the actual name of the database, not an alias.

If the argument is NULL or an empty string, metrics are taken from all active databases in the instance. This input argument applies only to database level memory sets.

member

An input argument of type INTEGER that specifies from which member the data is returned. Specify -1 for the current database member, or -2 for all active members. If the NULL value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

Example 1: Retrieve memory set metrics for the current instance and the currently connected database.

```
SELECT varchar(memory_set_type, 20) AS set_type,  
       varchar(memory_pool_type, 20) AS pool_type,  
       varchar(db_name, 20) AS dbname,  
       memory_pool_used,  
       memory_pool_used_hwm  
FROM TABLE(  
    MON_GET_MEMORY_POOL(NULL, CURRENT_SERVER, -2))
```

An example of output from this query.

| SET_TYPE | POOL_TYPE | DBNAME | MEMORY_POOL_USED | MEMORY_POOL_USED_HWM |
|-------------|----------------|--------|------------------|----------------------|
| DBMS | FCM_LOCAL | - | 0 | 0 |
| DBMS | FCM_SESSION | - | 2359296 | 2359296 |
| DBMS | FCM_CHANNEL | - | 589824 | 589824 |
| DBMS | FCMBP | - | 983040 | 983040 |
| DBMS | FCM_CHANNEL | - | 35520512 | 35520512 |
| DBMS | MONITOR | - | 458752 | 589824 |
| DBMS | RESYNC | - | 262144 | 262144 |
| DBMS | OSS_TRACKER | - | 7667712 | 7667712 |
| DBMS | APM | - | 13041664 | 13238272 |
| DBMS | BSU | - | 3932160 | 4390912 |
| DBMS | KERNEL_CONTROL | - | 3932160 | 4390912 |
| DBMS | EDU | - | 655360 | 655360 |
| FMP | MISC | - | 655360 | 655360 |
| DATABASE | UTILITY | TESTDB | 65536 | 65536 |
| DATABASE | PACKAGE_CACHE | TESTDB | 983040 | 983040 |
| DATABASE | XMLCACHE | TESTDB | 196608 | 196608 |
| DATABASE | CAT_CACHE | TESTDB | 458752 | 458752 |
| DATABASE | BP | TESTDB | 850132992 | 850132992 |
| DATABASE | BP | TESTDB | 655360 | 655360 |
| APPLICATION | APPLICATION | TESTDB | 393216 | 393216 |
| APPLICATION | APPLICATION | TESTDB | 262144 | 262144 |

21 record(s) selected

Usage notes

In a partitioned database environment, the fast communication manager (FCM) memory set is allocated per host; all members on this host machine share this set. The MON_GET_MEMORY_POOL function retrieves data from each member. Since the FCM memory set is shared among all members on the host, the metrics reported for FCM memory for each member on the host represent information about the same shared memory set. For this reason, when examining metrics for FCM memory, examine the data for each unique host. For hosts with multiple members, use data from only one member on that host, as the metrics for FCM memory represent the aggregated total for all members on the given host.

Information returned

Table 139. Information returned for MON_GET_MEMORY_POOL

| Column name | Data type | Description |
|-------------|-----------|--------------------------|
| MEMBER | SMALLINT | member - Database member |

Table 139. Information returned for `MON_GET_MEMORY_POOL` (continued)

| Column name | Data type | Description |
|----------------------|--------------|---|
| HOST_NAME | VARCHAR(255) | host_name - Host name |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| MEMORY_SET_TYPE | VARCHAR(32) | memory_set_type - Memory set type. See the <i>memory_set_type</i> input parameter for the list of possible types. |
| MEMORY_POOL_TYPE | VARCHAR(32) | memory_pool_type - Memory pool type. |
| MEMORY_POOL_ID | BIGINT | memory_pool_id - Memory pool identifier |
| APPLICATION_HANDLE | BIGINT | application_handle - Application handle. Only applicable to APPLICATION, STATISTICS, STATEMENT, and SORT_PRIVATE memory pool types. Otherwise, the value is NULL. |
| EDU_ID | BIGINT | edu_id - Engine dispatchable unit ID monitor element. Only applicable for memory pools allocated from the PRIVATE memory set type. Otherwise, the value is NULL. |
| MEMORY_POOL_USED | BIGINT | memory_pool_used - Amount of memory pool in use |
| MEMORY_POOL_USED_HWM | BIGINT | memory_pool_used_hwm - Memory pool high water mark |

MON_GET_MEMORY_SET - get memory set information

The `MON_GET_MEMORY_SET` table function retrieves metrics from the allocated memory sets, both at the instance level, and for all active databases within the instance.

Syntax

►► `MON_GET_MEMORY_SET` (*—memory_set_type—*, *—db_name—*, *—member—*) ◀◀

The schema is SYSPROC.

Table function parameters

memory_set_type

An input argument of type VARCHAR(32) that specifies the type of the memory set when calling this function. If the argument is NULL or an empty string, then metrics are retrieved for all memory sets at the instance and database level. Otherwise metrics for the specified memory set are retrieved.

These parameter values are accepted:

| Value | Scope | Description |
|-------------|----------|--------------------------------------|
| DBMS | Instance | Database manager (DBM) memory set |
| FMP | Instance | Fenced mode process (FMP) memory set |
| PRIVATE | Instance | Private memory set |
| DATABASE | Database | Database memory set |
| APPLICATION | Database | Application memory set |

| Value | Scope | Description |
|-------|--|--|
| FCM | Host - there is only one FCM memory set allocated per machine for an instance. | Fast communication manager (FCM) memory set |
| NULL | All | All memory sets at the instance and database level |

db_name

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database when calling this function.

The database must have a directory entry type of either “INDIRECT” or “HOME”, as returned by a LIST DATABASE DIRECTORY command. The database must be active. Alternatively, the CURRENT_SERVER special register can be specified to retrieve metrics from the currently connected database. The register value contains the actual name of the database, not an alias.

If the argument is NULL or an empty string, metrics are taken from all active databases in the instance. This input argument applies only to database level memory sets.

member

An input argument of type INTEGER that specifies from which member the data is returned. Specify -1 for the current database member, or -2 for all active members. If the NULL value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

Example 1: Retrieve memory set metrics for the current instance and the currently connected database.

```
SELECT varchar(memory_set_type, 20) as set_type,
       varchar(db_name, 20) as dbname,
       memory_set_used,
       memory_set_used_hwm
FROM TABLE(
    MON_GET_MEMORY_SET(NULL, CURRENT_SERVER, -2))
```

An example of output from this query.

| SET_TYPE | DBNAME | MEMORY_SET_USED | MEMORY_SET_USED_HWM |
|----------|--------|-----------------|---------------------|
| DBMS | - | 86080 | 87360 |
| FMP | - | 0 | 704 |
| PRIVATE | - | 10624 | 16256 |

| | | | |
|-------------|--------|--------|--------|
| DATABASE | TESTDB | 928000 | 928000 |
| APPLICATION | TESTDB | 1472 | 2752 |

5 record(s) selected

Usage notes

In a partitioned database environment, the fast communication manager (FCM) memory set is allocated per host; all members on this host machine share this set. The MON_GET_MEMORY_SET function retrieves data from each member. Since the FCM memory set is shared among all members on the host, the metrics reported for FCM memory for each member on the host represent information about the same shared memory set. For this reason, when examining metrics for FCM memory, examine the data for each unique host. For hosts with multiple members, use data from only one member on that host, as the metrics for FCM memory represent the aggregated total for all members on the given host.

Information returned

Table 140. Information returned for MON_GET_MEMORY_SET

| Column name | Data type | Description |
|----------------------|--------------|---|
| MEMBER | SMALLINT | member - Database member |
| HOST_NAME | VARCHAR(255) | host_name - Host name |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| MEMORY_SET_TYPE | VARCHAR(32) | memory_set_type - Memory set type. See the <i>memory_set_type</i> input parameter for the list of possible types. |
| MEMORY_SET_ID | BIGINT | memory_set_id - Memory set identifier. |
| MEMORY_SET_SIZE | BIGINT | memory_set_size - Memory set size. The value is in KB. |
| MEMORY_SET_COMMITTED | BIGINT | memory_set_committed - Memory currently committed. The value is in KB. |
| MEMORY_SET_USED | BIGINT | memory_set_used - Memory in use by this set. The value is in KB. |
| MEMORY_SET_USED_HWM | BIGINT | memory_set_used_hwm - Memory set high water mark. The value is in KB. |

MON_GET_PAGE_ACCESS_INFO table function - Get buffer pool page waiting information

The MON_GET_PAGE_ACCESS_INFO table function returns information about bufferpool pages that are being waited on for a specified table. This is only applicable to Db2 pureScale instances.

Syntax

►►—MON_GET_PAGE_ACCESS_INFO—(—*tabschema*—,—*tabname*—,—*member*—)—►►

The schema is SYSPROC.

Table function parameters

tabschema

An input argument of type VARCHAR(128) that specifies the database schema to query. If the argument is null or an empty string, information for all schemas is returned.

tabname

An input argument of type VARCHAR(128) that specifies the table name to query. If the argument is null or an empty string, information for all tables is returned.

member

An input argument of type INTEGER that specifies a valid member number in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

This example returns page reclaim counts for all tables in schema BASETAB on the currently connected member. It shows that applications are waiting for pages for table TABLE1 (an example of what could cause this situation is updating different rows on the same page from two different members).

```
SELECT SUBSTR(TABNAME,1,8) AS NAME,
       SUBSTR(OBJTYPE,1,5) AS TYPE,
       PAGE_RECLAIMS_X AS PGRCX,
       PAGE_RECLAIMS_S AS PGRCS,
       SPACEMAPPAGE_PAGE_RECLAIMS_X AS SMPPGRCX,
       SPACEMAPPAGE_PAGE_RECLAIMS_S AS SMPPGRCS
FROM TABLE( MON_GET_PAGE_ACCESS_INFO('BASETAB', NULL, NULL) ) AS WAITMETRICS

ORDER BY NAME;
```

The following is an example of output from this query.

| NAME | TYPE | PGRCX | PGRCS | SMPPGRCX | SMPPGRCS |
|--------|-------|-------|-------|----------|----------|
| TABLE1 | TABLE | 12641 | 320 | 72 | 17 |
| TABLE1 | INDEX | 5042 | 78 | 7 | 0 |
| TABLE2 | TABLE | 420 | 12 | 9 | 0 |
| TABLE2 | INDEX | 7 | 0 | 0 | 0 |

Usage notes

The MON_GET_PAGE_ACCESS_INFO table function returns one or two rows of data for each table per member that it gathers data from. One row of data shows

information for INDEX pages. The second row shows information for DATA pages. No aggregation across members is performed.

Information returned

Table 141. Information returned for MON_GET_PAGE_ACCESS_INFO

| Column name | Data type | Description or corresponding monitor element |
|--|--------------|--|
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TABNAME | VARCHAR(128) | table_name - Table name |
| OBJTYPE | VARCHAR(128) | Type of object. One of: <ul style="list-style-type: none"> • INDEX • TABLE |
| MEMBER | SMALLINT | member - Database member |
| PAGE_RECLAIMS_X | BIGINT | page_reclaims_x - Page reclaims exclusive access |
| PAGE_RECLAIMS_S | BIGINT | page_reclaims_s - Page reclaims shared access |
| SPACEMAPPAGE_PAGE_RECLAIMS_X | BIGINT | spacemappage_page_reclaims_x - Space map page reclaims exclusive access |
| SPACEMAPPAGE_PAGE_RECLAIMS_S | BIGINT | spacemappage_page_reclaims_s - Space map page reclaims shared access |
| PAGE_RECLAIMS_INITIATED_X | BIGINT | page_reclaims_initiated_x - Page reclaims initiated exclusive access |
| PAGE_RECLAIMS_INITIATED_S | BIGINT | page_reclaims_initiated_s - Page reclaims initiated shared access |
| SPACEMAPPAGE_PAGE_RECLAIMS_INITIATED_X | BIGINT | spacemappage_page_reclaims_initiated_x - Space map page reclaims initiated exclusive access monitor element |
| SPACEMAPPAGE_PAGE_RECLAIMS_INITIATED_S | BIGINT | spacemappage_page_reclaims_initiated_s - Space map page reclaims initiated shared access monitor element |
| RECLAIM_WAIT_TIME | BIGINT | reclaim_wait_time - Reclaim wait time |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME | BIGINT | spacemappage_reclaim_wait_time - Space map page reclaim wait time |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data partition identifier |
| IID | SMALLINT | iid - Index identifier. This monitor element only returns a value when the OBJTYPE is INDEX and the index is a nonpartitioned type on a partitioned table. Otherwise, the value of this monitor element is NULL. |

MON_GET_PKG_CACHE_STMT table function - Get package cache statement metrics

The MON_GET_PKG_CACHE_STMT table function returns a point-in-time view of both static and dynamic SQL statements in the database package cache.

Syntax

```

►►—MON_GET_PKG_CACHE_STMT—(—section_type—, —————►
►—executable_id—, —search_args—, —member—)—————►►

```

The schema is SYSPROC.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Routine parameters

section_type

An optional input argument (either "D" or "S") of type CHAR(1) that specifies information type for the returned statement. If the argument is NULL or the empty string, information is returned for all SQL statements. Not case-sensitive: "D" stands for dynamic; "S" for static.

executable_id

An optional input argument of type VARCHAR (32) for bit data that specifies a unique section of the database package cache. If a null value is specified, information is returned for all SQL statements. When the *executable_id* is specified, the *section_type* argument is ignored. For example, if an *executable_id* is specified for a dynamic statement, the dynamic statement details will be returned by this table function even if *section_type* is specified as static ("S").

search_args

An optional input parameter of type CLOB(1K), that allows you to specify one or more optional search argument strings. For example:

```
'<modified_within>5</modified_within><update_boundary_time>myPkgEvmon
  </update_boundary_time>'
```

The available search argument tags are as follows:

- '<modified_within>X</modified_within>'

Returns only those statement entries that have either been inserted into the cache or executed within the last X minutes (where X is a positive integer value). If the argument is not specified, all entries in the cache are returned.

- '<update_boundary_time>evmon_name</update_boundary_time>'

Updates the event monitor boundary timestamp to the current time for the package cache event monitor specified by *evmon_name*. If this event monitor specifies where updated_since_boundary_time as an output criteria in its WHERE clause, only package cache entries that subsequently have their metrics updated are captured when evicted from the package cache. This operation only has an effect if the specified package cache event monitor is active when the command is issued.

Each input argument can be specified only once. The search argument tags must be specified in lowercase.

member

An optional input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this

function. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set.

Information returned

Table 142. Information returned for MON_GET_PKG_CACHE_STMT

| Column Name | Data Type | Description or corresponding monitor element |
|------------------------------|--------------------------|--|
| MEMBER | SMALLINT | member - Database member |
| SECTION_TYPE | CHAR(1) | section_type - Section type indicator. |
| INSERT_TIMESTAMP | TIMESTAMP | insert_timestamp - Statement insert timestamp |
| EXECUTABLE_ID | VARCHAR(32) FOR BIT DATA | executable_id - Executable ID. |
| PACKAGE_SCHEMA | VARCHAR(128) | package_schema - Package schema. This output is valid for static SQL statements only. A NULL value is returned if the statement is dynamic. |
| PACKAGE_NAME | VARCHAR(128) | package_name - Package name. This output is valid for static SQL statements only. A NULL value is returned if the statement is dynamic. |
| PACKAGE_VERSION_ID | VARCHAR(64) | package_version_id - Package version. This output is valid for static SQL statements only. A NULL value is returned if the statement is dynamic or if you did not specify the package version for static statement. An empty string will be returned for static statement if the package version identifier was not specified by you when the package was created. |
| SECTION_NUMBER | BIGINT | section_number - Section number. A NULL value is returned if the statement is dynamic. |
| EFFECTIVE_ISOLATION | CHAR(2) | effective_isolation - Effective isolation. This is the isolation value in effect for the section; it can be different from what it was originally requested at compilation time. |
| NUM_EXECUTIONS | BIGINT | num_executions - Statement executions |
| NUM_EXEC_WITH_METRICS | BIGINT | num_exec_with_metrics - Number of executions with metrics collected. |
| PREP_TIME | BIGINT | prep_time - Preparation time Note that PREP_TIME is only valid for dynamic SQL statements. PREP_TIME is reported as 0 for static SQL statements. |
| TOTAL_ACT_TIME | BIGINT | total_act_time - Total activity time |
| TOTAL_ACT_WAIT_TIME | BIGINT | total_act_wait_time - Total activity wait time |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| TOTAL_SECTION_SORT_TIME | BIGINT | total_section_sort_time - Total section sort time |
| TOTAL_SECTION_SORT_PROC_TIME | BIGINT | total_section_sort_proc_time - Total section sort processing time |
| TOTAL_SECTION_SORTS | BIGINT | total_section_sorts - Total section sorts |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| ROWS_MODIFIED | BIGINT | rows_modified - Rows modified |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_RETURNED | BIGINT | rows_returned - Rows returned |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |

Table 142. Information returned for MON_GET_PKG_CACHE_STMT (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|-----------------------------|-----------|--|
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer pool XDA data physical reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer pool XDA data writes |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| TOTAL_SORTS | BIGINT | total_sorts - Total Sorts |
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post threshold sorts |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post shared threshold sorts |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| WLM_QUEUE_TIME_TOTAL | BIGINT | wlm_queue_time_total - Workload manager total queue time |
| WLM_QUEUE_ASSIGNMENTS_TOTAL | BIGINT | wlm_queue_assignments_total - Workload manager total queue assignments |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |
| FCM_RECV_VOLUME | BIGINT | fcm_recv_volume - FCM recv volume |
| FCM_RECVS_TOTAL | BIGINT | fcm_recv_total - FCM recvs total |
| FCM_SEND_VOLUME | BIGINT | fcm_send_volume - FCM send volume |
| FCM_SENDS_TOTAL | BIGINT | fcm_sends_total - FCM sends total |
| FCM_RECV_WAIT_TIME | BIGINT | fcm_recv_wait_time - FCM recv wait time |
| FCM_SEND_WAIT_TIME | BIGINT | fcm_send_wait_time - FCM send wait time |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |
| LOG_BUFFER_WAIT_TIME | BIGINT | log_buffer_wait_time - Log buffer wait time |
| NUM_LOG_BUFFER_FULL | BIGINT | num_log_buffer_full - Number of full log buffers |
| LOG_DISK_WAIT_TIME | BIGINT | log_disk_wait_time - Log disk wait time |
| LOG_DISK_WAITS_TOTAL | BIGINT | log_disk_waits_total - Log disk waits total |
| LAST_METRICS_UPDATE | TIMESTAMP | last_metrics_update - Metrics last update timestamp |
| NUM_COORD_EXEC | BIGINT | num_coord_exec - Number of executions by coordinator agent |

Table 142. Information returned for MON_GET_PKG_CACHE_STMT (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|----------------------------------|-------------|--|
| NUM_COORD_EXEC_WITH_METRICS | BIGINT | num_coord_exec_with_metrics - Number of executions by coordinator agent |
| VALID | CHAR(1) | valid - Section validity indicator. |
| TOTAL_ROUTINE_TIME | BIGINT | total_routine_time - Total routine time |
| TOTAL_ROUTINE_INVOCATIONS | BIGINT | total_routine_invocations - Total routine invocations |
| ROUTINE_ID | BIGINT | routine_id - Routine ID monitor element |
| STMT_TYPE_ID | VARCHAR(32) | stmt_type_id - Statement type identifier |
| QUERY_COST_ESTIMATE | BIGINT | query_cost_estimate - Query cost estimate |
| STMT_PKG_CACHE_ID | BIGINT | stmt_pkgcache_id - Statement package cache identifier |
| COORD_STMT_EXEC_TIME | BIGINT | coord_stmt_exec_time - Execution time for statement by coordinator agent |
| STMT_EXEC_TIME | BIGINT | stmt_exec_time - Statement execution time |
| TOTAL_SECTION_TIME | BIGINT | total_section_time - Total section time |
| TOTAL_SECTION_PROC_TIME | BIGINT | total_section_proc_time - Total section processing time |
| TOTAL_ROUTINE_NON_SECT_TIME | BIGINT | total_routine_non_sect_time - Non-section routine execution time |
| TOTAL_ROUTINE_NON_SECT_PROC_TIME | BIGINT | total_routine_non_sect_proc_time - Non-section processing time |
| LOCK_WAITS_GLOBAL | BIGINT | lock_waits_global - Lock waits global |
| LOCK_WAIT_TIME_GLOBAL | BIGINT | lock_wait_time_global - Lock wait time global |
| LOCK_TIMEOUTS_GLOBAL | BIGINT | lock_timeouts_global - Lock timeouts global |
| LOCK_ESCALS_MAXLOCKS | BIGINT | lock_escals_maxlocks - Number of maxlocks lock escalations |
| LOCK_ESCALS_LOCKLIST | BIGINT | lock_escals_locklist - Number of locklist lock escalations |
| LOCK_ESCALS_GLOBAL | BIGINT | lock_escals_global - Number of global lock escalations |
| RECLAIM_WAIT_TIME | BIGINT | reclaim_wait_time - Reclaim wait time |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME | BIGINT | spacemappage_reclaim_wait_time - Space map page reclaim wait time |
| CF_WAITS | BIGINT | cf_waits - Number of cluster caching facility waits |
| CF_WAIT_TIME | BIGINT | cf_wait_time - cluster caching facility wait time |
| POOL_DATA_GBP_L_READS | BIGINT | pool_data_gbp_l_reads - Group buffer pool data logical reads |
| POOL_DATA_GBP_P_READS | BIGINT | pool_data_gbp_p_reads - Group buffer pool data physical reads |
| POOL_DATA_LBP_PAGES_FOUND | BIGINT | pool_data_lbp_pages_found - Local buffer pool found data pages |
| POOL_DATA_GBP_INVALID_PAGES | BIGINT | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages |
| POOL_INDEX_GBP_L_READS | BIGINT | pool_index_gbp_l_reads - Group buffer pool index logical reads |
| POOL_INDEX_GBP_P_READS | BIGINT | pool_index_gbp_p_reads - Group buffer pool index physical reads |
| POOL_INDEX_LBP_PAGES_FOUND | BIGINT | pool_index_lbp_pages_found - Local buffer pool index pages found |
| POOL_INDEX_GBP_INVALID_PAGES | BIGINT | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages |
| POOL_XDA_GBP_L_READS | BIGINT | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |
| POOL_XDA_GBP_P_READS | BIGINT | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |

Table 142. Information returned for MON_GET_PKG_CACHE_STMT (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|-----------------------------------|-------------|---|
| POOL_XDA_LBP_PAGES_FOUND | BIGINT | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |
| POOL_XDA_GBP_INVALID_PAGES | BIGINT | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| AUDIT_EVENTS_TOTAL | BIGINT | audit_events_total - Total audit events |
| AUDIT_FILE_WRITES_TOTAL | BIGINT | audit_file_writes_total - Total Audit files written |
| AUDIT_FILE_WRITE_WAIT_TIME | BIGINT | audit_file_write_wait_time - Audit file write wait time |
| AUDIT_SUBSYSTEM_WAITS_TOTAL | BIGINT | audit_subsystem_waits_total - Total audit subsystem waits |
| AUDIT_SUBSYSTEM_WAIT_TIME | BIGINT | audit_subsystem_wait_time - Audit subsystem wait time |
| DIAGLOG_WRITES_TOTAL | BIGINT | diaglog_writes_total - Diag log total writes |
| DIAGLOG_WRITE_WAIT_TIME | BIGINT | diaglog_write_wait_time - Diag log write time |
| FCM_MESSAGE_RECVS_TOTAL | BIGINT | fcm_message_recvs_total - FCM message recvs total |
| FCM_MESSAGE_RECV_VOLUME | BIGINT | fcm_message_recv_volume - FCM message recv volume |
| FCM_MESSAGE_RECV_WAIT_TIME | BIGINT | fcm_message_recv_wait_time - FCM message recv wait time |
| FCM_MESSAGE_SENDS_TOTAL | BIGINT | fcm_message_sends_total - FCM message sends total |
| FCM_MESSAGE_SEND_VOLUME | BIGINT | fcm_message_send_volume - FCM message send volume |
| FCM_MESSAGE_SEND_WAIT_TIME | BIGINT | fcm_message_send_wait_time - FCM message send wait time |
| FCM_TQ_RECVS_TOTAL | BIGINT | fcm_tq_recvs_total - FCM tablequeue recvs total |
| FCM_TQ_RECV_VOLUME | BIGINT | fcm_tq_recv_volume - FCM tablequeue recv volume |
| FCM_TQ_RECV_WAIT_TIME | BIGINT | fcm_tq_recv_wait_time - FCM tablequeue recv wait time |
| FCM_TQ_SENDS_TOTAL | BIGINT | fcm_tq_sends_total - FCM tablequeue send total |
| FCM_TQ_SEND_VOLUME | BIGINT | fcm_tq_send_volume - FCM tablequeue send volume |
| FCM_TQ_SEND_WAIT_TIME | BIGINT | fcm_tq_send_wait_time - FCM tablequeue send wait time |
| NUM_LW_THRESH_EXCEEDED | BIGINT | num_lw_thresh_exceeded - Number of thresholds exceeded |
| THRESH_VIOLATIONS | BIGINT | thresh_violations - Number of threshold violations |
| TOTAL_APP_SECTION_EXECUTIONS | BIGINT | total_app_section_executions - Total section executions |
| TOTAL_ROUTINE_USER_CODE_PROC_TIME | BIGINT | total_routine_user_code_proc_time - Total routine user code processing time |
| TOTAL_ROUTINE_USER_CODE_TIME | BIGINT | total_routine_user_code_time - Total routine user code time |
| TQ_TOT_SEND_SPILLS | BIGINT | tq_tot_send_spills - Total number of table queue buffers overflowed |
| EVMON_WAIT_TIME | BIGINT | evmon_wait_time - Event monitor wait time |
| EVMON_WAITS_TOTAL | BIGINT | evmon_waits_total - Event monitor total waits |
| TOTAL_EXTENDED_LATCH_WAIT_TIME | BIGINT | total_extended_latch_wait_time - Total extended latch wait time |
| TOTAL_EXTENDED_LATCH_WAITS | BIGINT | total_extended_latch_waits - Total extended latch waits |
| MAX_COORD_STMT_EXEC_TIME | BIGINT | max_coord_stmt_exec_time - Maximum coordinator statement execution time |
| MAX_COORD_STMT_EXEC_TIMESTAMP | TIMESTAMP | max_coord_stmt_exec_timestamp - Maximum coordinator statement execution timestamp monitor element |
| TOTAL_DISP_RUN_QUEUE_TIME | BIGINT | total_disp_run_queue_time - Total dispatcher run queue time |
| QUERY_DATA_TAG_LIST | VARCHAR(32) | query_data_tag_list - Query data tag list |
| TOTAL_STATS_FABRICATION_TIME | BIGINT | total_stats_fabrication_time - Total statistics fabrication time |
| TOTAL_STATS_FABRICATIONS | BIGINT | total_stats_fabrications - Total statistics fabrications |

Table 142. Information returned for MON_GET_PKG_CACHE_STMT (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|------------------------------------|-----------|--|
| TOTAL_SYNC_RUNSTATS_TIME | BIGINT | total_sync_runstats_time - Total synchronous RUNSTATS time |
| TOTAL_SYNC_RUNSTATS | BIGINT | total_sync_runstats - Total synchronous RUNSTATS activities |
| TOTAL_PEDS | BIGINT | total_peds - Total partial early distincts |
| DISABLED_PEDS | BIGINT | disabled_peds - Disabled partial early distincts |
| POST_THRESHOLD_PEDS | BIGINT | post_threshold_peds - Partial early distincts threshold |
| TOTAL_PEAS | BIGINT | total_peas - Total partial early aggregations |
| POST_THRESHOLD_PEAS | BIGINT | post_threshold_peas - Partial early aggregation threshold |
| TQ_SORT_HEAP_REQUESTS | BIGINT | tq_sort_heap_requests - Table queue sort heap requests |
| TQ_SORT_HEAP_REJECTIONS | BIGINT | tq_sort_heap_rejections - Table queue sort heap rejections |
| POOL_QUEUED_ASYNC_DATA_REQS | BIGINT | pool_queued_async_data_reqs - Data prefetch requests monitor element |
| POOL_QUEUED_ASYNC_INDEX_REQS | BIGINT | pool_queued_async_index_reqs - Index prefetch requests monitor element |
| POOL_QUEUED_ASYNC_XDA_REQS | BIGINT | pool_queued_async_xda_reqs - XDA prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_OTHER_REQS | BIGINT | pool_queued_async_other_reqs - Non-prefetch requests monitor element |
| POOL_QUEUED_ASYNC_DATA_PAGES | BIGINT | pool_queued_async_data_pages - Data pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_INDEX_PAGES | BIGINT | pool_queued_async_index_pages - Index pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_XDA_PAGES | BIGINT | pool_queued_async_xda_pages - XDA pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES | BIGINT | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES | BIGINT | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_DATA_REQS | BIGINT | pool_failed_async_data_reqs - Failed data prefetch requests monitor element |
| POOL_FAILED_ASYNC_INDEX_REQS | BIGINT | pool_failed_async_index_reqs - Failed index prefetch requests monitor element |
| POOL_FAILED_ASYNC_XDA_REQS | BIGINT | pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element |
| POOL_FAILED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element |

Table 142. Information returned for MON_GET_PKG_CACHE_STMT (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|--|
| POOL_FAILED_ASYNC_OTHER_REQS | BIGINT | pool_failed_async_other_reqs - Failed non-prefetch requests monitor element |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| PREFETCH_WAITS | BIGINT | prefetch_waits - Prefetcher wait count monitor element |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |
| NUM_WORKING_COPIES | BIGINT | Number of working copies. |
| IDA_SEND_WAIT_TIME | BIGINT | ida_send_wait_time - Time spent waiting to send data monitor element

The total amount of time spent waiting to send data to an in-database analytics process. |
| IDA_SENDS_TOTAL | BIGINT | ida_sends_total - Number of times data sent monitor element

The total number of times data was sent to an in-database analytics process. |
| IDA_SEND_VOLUME | BIGINT | ida_send_volume - Total data volume sent monitor element

The total volume of data sent from the database server to an in-database analytics process. The value is reported in bytes. |
| IDA_RECV_WAIT_TIME | BIGINT | ida_recv_wait_time - Time spent waiting to receive data monitor element

The total amount of time spent waiting to receive data from an in-database analytics process. |
| IDA_RECVS_TOTAL | BIGINT | ida_recv_total - Number of times data received monitor element

The total number of times data was received from an in-database analytics process. |
| IDA_RECV_VOLUME | BIGINT | ida_recv_volume - Total data volume received monitor element

The total volume of data the database server received from an in-database analytics process. The value is reported in bytes. |
| STMTNO | INTEGER | stmtno - Statement number monitor element |
| NUM_ROUTINES | INTEGER | num_routines - Number of routines |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted monitor element |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted monitor element |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated monitor element |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total Hash Joins monitor element |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total Hash Loops monitor element |
| HASH_JOIN_OVERFLOWS | BIGINT | hash_join_overflows - Hash Join Overflows monitor element |

Table 142. Information returned for MON_GET_PKG_CACHE_STMT (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---------------------------------------|-----------|---|
| HASH_JOIN_SMALL_OVERFLOWS | BIGINT | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| POST_SHRTHRESHOLD_HASH_JOINS | BIGINT | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP Functions monitor element |
| OLAP_FUNC_OVERFLOWS | BIGINT | olap_func_overflows - OLAP Function Overflows monitor element |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal Rows Inserted monitor element |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal Rows Updated monitor element |
| POOL_COL_L_READS | BIGINT | pool_col_l_reads - Buffer pool column-organized logical reads |
| POOL_TEMP_COL_L_READS | BIGINT | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| POOL_COL_P_READS | BIGINT | pool_col_p_reads - Buffer pool column-organized physical reads |
| POOL_TEMP_COL_P_READS | BIGINT | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| POOL_COL_LBP_PAGES_FOUND | BIGINT | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found |
| POOL_COL_WRITES | BIGINT | pool_col_writes - Buffer pool column-organized writes |
| POOL_COL_GBP_L_READS | BIGINT | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| POOL_COL_GBP_P_READS | BIGINT | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| POOL_COL_GBP_INVALID_PAGES | BIGINT | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |
| POOL_QUEUED_ASYNC_COL_REQS | BIGINT | pool_queued_async_col_reqs - Column-organized prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_REQS | BIGINT | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| POOL_QUEUED_ASYNC_COL_PAGES | BIGINT | pool_queued_async_col_pages - Column-organized page prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_PAGES | BIGINT | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| POOL_FAILED_ASYNC_COL_REQS | BIGINT | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| POOL_FAILED_ASYNC_TEMP_COL_REQS | BIGINT | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |
| TOTAL_COL_TIME | BIGINT | total_col_time - Total column-organized time |
| TOTAL_COL_PROC_TIME | BIGINT | total_col_proc_time - Total column-organized processing time |
| TOTAL_COL_EXECUTIONS | BIGINT | total_col_executions - Total column-organized executions |
| COMM_EXIT_WAIT_TIME | BIGINT | comm_exit_wait_time - Communication buffer exit wait time |
| COMM_EXIT_WAITS | BIGINT | comm_exit_waits - Communication buffer exit number of waits |

Table 142. Information returned for MON_GET_PKG_CACHE_STMT (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|---|
| POST_THRESHOLD_HASH_JOINS | BIGINT | post_threshold_hash_joins - Hash Join Threshold monitor element |
| POOL_DATA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_READ_TIME | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_WRITE_TIME | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |

Table 142. Information returned for MON_GET_PKG_CACHE_STMT (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|---|
| POOL_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| TOTAL_HASH_GRPBYS | BIGINT | total_hash_grpbys - Total hash group by operations |
| HASH_GRPBY_OVERFLOWS | BIGINT | hash_grby_overflows - Hash group by overflows |
| POST_THRESHOLD_HASH_GRPBYS | BIGINT | post_threshold_hash_grpbys - Hash group by threshold |
| POST_THRESHOLD_OLAP_FUNCS | BIGINT | post_threshold_olap_funcs - OLAP function threshold monitor element |
| EXT_TABLE_RECV_WAIT_TIME | BIGINT | ext_table_rcv_wait_time - Total agent wait time for external table readers monitor element |
| EXT_TABLE_RECVS_TOTAL | BIGINT | ext_table_rcvs_total - Total row batches received from external table readers monitor element |
| EXT_TABLE_RECV_VOLUME | BIGINT | ext_table_rcv_volume - Total data received from external table readers monitor element |
| EXT_TABLE_READ_VOLUME | BIGINT | ext_table_read_volume - Total data read by external table readers monitor element |
| EXT_TABLE_SEND_WAIT_TIME | BIGINT | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| EXT_TABLE_SENDS_TOTAL | BIGINT | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| EXT_TABLE_SEND_VOLUME | BIGINT | ext_table_send_volume - Total data sent to external table writers monitor element |
| EXT_TABLE_WRITE_VOLUME | BIGINT | ext_table_write_volume - Total data written by external table writers monitor element |
| SEMANTIC_ENV_ID | BIGINT | semantic_env_id - Query semantic compilation environment ID monitor element |
| STMTID | BIGINT | stmtid - Query statement ID monitor element |
| PLANID | BIGINT | planid - Query plan ID monitor element |
| PREP_WARNING | INTEGER | prep_warning - Prepare warning SQLCODE monitor element |
| PREP_WARNING_REASON | INTEGER | prep_warning_reason - Prepare warning SQLCODE reason identifier monitor element |
| POST_THRESHOLD_COL_VECTOR_CONSUMERS | BIGINT | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| TOTAL_COL_VECTOR_CONSUMERS | BIGINT | total_col_vector_consumers - Total columnar_vector consumers |
| ACTIVE_HASH_GRPBYS_TOP | BIGINT | active_hash_grpbys_top - Active hash Group By operations high watermark |
| ACTIVE_HASH_JOINS_TOP | BIGINT | active_hash_joins_top - Active hash joins operations high watermark |
| ACTIVE_OLAP_FUNCS_TOP | BIGINT | active_olap_funcs_top - Active OLAP functions operations high watermark |
| ACTIVE_PEAS_TOP | BIGINT | active_peas_top - Active partial early aggregations operations high watermark |
| ACTIVE_PEDS_TOP | BIGINT | active_peds_top - Active partial early distinct operations high watermark |
| ACTIVE_SORT_CONSUMERS_TOP | BIGINT | active_sort_consumers_top - Active sort consumers high watermark |
| ACTIVE_SORTS_TOP | BIGINT | active_sorts_top - Active Sorts high watermark |
| ACTIVE_COL_VECTOR_CONSUMERS_TOP | BIGINT | active_col_vector_consumers_top - Active columnar_vector consumers high watermark |
| SORT_CONSUMER_HEAP_TOP | BIGINT | sort_consumer_heap_top - Individual private sort heap consumer high watermark |

Table 142. Information returned for MON_GET_PKG_CACHE_STMT (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|--------------------------------|-----------|--|
| SORT_CONSUMER_SHRHEAP_TOP | BIGINT | sort_consumer_shrheap_top - Individual shared sort heap consumer high watermark |
| SORT_HEAP_TOP | BIGINT | sort_heap_top - Sort private heap high watermark |
| SORT_SHRHEAP_TOP | BIGINT | sort_shrheap_top - Sort share heap high watermark |
| TOTAL_INDEX_BUILD_TIME | BIGINT | total_index_build_time - Total time spent building indexes due to index creation |
| TOTAL_INDEX_BUILD_PROC_TIME | BIGINT | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation |
| TOTAL_INDEXES_BUILT | BIGINT | total_indexes_built - Total number of indexes built |
| FCM_TQ_RECV_WAITS_TOTAL | BIGINT | fcm_tq_recv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| FCM_MESSAGE_RECV_WAITS_TOTAL | BIGINT | fcm_message_recv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| FCM_TQ_SEND_WAITS_TOTAL | BIGINT | fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |
| FCM_MESSAGE_SEND_WAITS_TOTAL | BIGINT | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| FCM_SEND_WAITS_TOTAL | BIGINT | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| FCM_RECV_WAITS_TOTAL | BIGINT | fcm_recv_waits_total - Number of times spent waiting to receive data through FCM monitor element |
| COL_VECTOR_CONSUMER_OVERFLOWES | BIGINT | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |
| TOTAL_COL_SYNOPSIS_TIME | BIGINT | total_col_synopsis_time - Total column-organized synopsis table time monitor element |
| TOTAL_COL_SYNOPSIS_PROC_TIME | BIGINT | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| TOTAL_COL_SYNOPSIS_EXECUTIONS | BIGINT | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |
| COL_SYNOPSIS_ROWS_INSERTED | BIGINT | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| ADM_OVERFLOWES | BIGINT | Reserved for future use. |
| ADM_BYPASS_ACT_TOTAL | BIGINT | Reserved for future use. |
| LOB_PREFETCH_WAIT_TIME | BIGINT | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| LOB_PREFETCH_REQS | BIGINT | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| FED_ROWS_DELETED | BIGINT | fed_rows_deleted - Rows deleted by a federation system monitor element |
| FED_ROWS_INSERTED | BIGINT | fed_rows_inserted - Rows inserted by a federation system monitor element |
| FED_ROWS_UPDATED | BIGINT | fed_rows_updated - Rows updated by a federation system monitor element |
| FED_ROWS_READ | BIGINT | fed_rows_read - Rows read by a federation system monitor element |
| FED_WAIT_TIME | BIGINT | fed_wait_time - Time spent by a federation server monitor element |
| FED_WAITS_TOTAL | BIGINT | fed_waits_total - Total number of execution times for a federation server monitor element |
| STMT_TEXT | CLOB(2M) | stmt_text - SQL statement text |

Table 142. Information returned for `MON_GET_PKG_CACHE_STMT` (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|-------------------------------|-----------|---|
| COMP_ENV_DESC | BLOB(10K) | comp_env_desc - Compilation environment handle. You can use the existing <code>COMPILATION_ENV</code> table function to get the detailed compilation environment of the specific statement if needed. |
| MAX_COORD_STMT_EXEC_TIME_ARGS | BLOB(10M) | max_coord_stmt_exec_time_args - Maximum coordinator statement execution time arguments |
| STMT_COMMENTS | BLOB(10K) | stmt_comments - Comments in SQL statement text monitor element |
| ESTIMATED_SORT_SHRHEAP_TOP | BIGINT | Reserved for future use. |
| ESTIMATED_SORT_CONSUMERS_TOP | BIGINT | Reserved for future use. |
| ESTIMATED_RUNTIME | BIGINT | Reserved for future use. |

Usage notes

The `MON_GET_PKG_CACHE_STMT` table function returns a point-in-time view of both static and dynamic SQL statements in the database package cache. This allows you to examine the aggregated metrics for a particular SQL statement, allowing you to quickly determine the reasons for poor query performance. The metrics returned are aggregates of the metrics gathered during each execution of the statement.

It also allows you to compare the behavior of an individual cached section, relative to the other statements, to assist in identifying the most expensive section or statements (in terms of the execution costs).

The activity metrics reported by this function are rolled up to the global package cache at the end of the execution of the activity.

Metrics collection for the execution of any statement is controlled through the `COLLECT ACTIVITY METRICS` clause on workloads, or the `mon_act_metrics` database configuration parameter at the database level. Metrics are only collected for executions of the statement if the statement was submitted by a connection associated with a workload or database for which activity metrics are enabled. The `num_exec_with_metrics` element returned by the `MON_GET_PKG_CACHE_STMT` function indicates how many executions of the statement have had metrics collected and have contributed to the aggregate metrics reported. If no metrics are collected for any execution of the statement, then the `num_exec_with_metrics` element is 0 and all metric values are returned as 0.

The output that is returned by `MON_GET_PKG_CACHE_STMT` is not directly comparable to the output obtained through the dynamic SQL snapshot. The following are differences between the two interfaces:

- `MON_GET_PKG_CACHE_STMT` reports data per individual section in the cache, whereas the dynamic SQL snapshot aggregates all sections for the statement. Therefore, `MON_GET_PKG_CACHE` provides more granularity. It distinguishes statements that have separate entries in the cache because of a combination of environment differences.
- `MON_GET_PKG_CACHE_STMT` captures the metrics during query plan execution. The dynamic SQL snapshot might include some of the work that is done when the section is loaded, depending on how the statement is run from the application side.

It takes a longer time period to build the compilation environment and to transfer statement text (which can be as large as 2 MB) between members. To improve performance when retrieving a list of all the statements from the package cache, do not select the STMT_TEXT and the COMP_ENV_DESC columns.

List all the dynamic SQL statements from the database package cache ordered by the average CPU time.

The following is an example of output from this query.

5 record(s) selected.

```
STMT_TEXT
-----
SELECT * FROM EMPLOYEE
```

all metrics are returned as 0 and the value of `num_exec_with_metrics` is also 0. Then Alex executes the same statement afterwards, but the metrics are collected this time for the execution of the statement and `num_exec_with_metrics` increments. So, after Brent and Alex execute that statement, the result of this query:

shows that the SELECT statement ran twice and one of the execution times had the activity metrics collected.

```

NUM_EXECUTIONS NUM_EXEC_WITH_METRICS STMT_TEXT
-----
                2                      1 SELECT count(*) FROM syscat.tables

1 record(s) selected.

```

MON_GET_PKG_CACHE_STMT_DETAILS table function - Get package cache statement metrics as an XML document

The MON_GET_PKG_CACHE_STMT_DETAILS table function returns metrics for one or more package cache entries.

The metrics returned by the MON_GET_PKG_CACHE_STMT_DETAILS table function represent the accumulation of all metrics for statements in the package cache. Statement metrics are rolled up to the package cache upon activity completion.

The metrics are returned in an XML document.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

```

►►—MON_GET_PKG_CACHE_STMT_DETAILS—(—section_type—,—————►
►—executable_id—, —search_args—, —member—)—————►►

```

The schema is SYSPROC.

Routine parameters

section_type

An optional input argument (either "D" or "S") of type CHAR(1) that specifies information type for the returned statement. If the argument is NULL or an empty string, information is returned for all SQL statements. Not case sensitive: D stands for dynamic; S for static.

executable_id

An optional input argument of type VARCHAR (32) for bit data that specifies a unique section of the database package cache. If a null value is specified, information is returned for all SQL statements. When the *executable_id* is specified, the *section_type* argument is ignored. For example, if an *executable_id* is specified for a dynamic statement, the dynamic statement details will be returned by this table function even if *section_type* is specified as static ("S").

search_args

An optional input parameter of type CLOB(1K), that allows you to specify one or more optional search argument strings. For example:

```
'<modified_within>5</modified_within><update_boundary_time>myPkgEvmon
  </update_boundary_time>'
```

The available search argument tags are as follows:

- '`<modified_within>X</modified_within>`'

Returns only those statement entries that have either been inserted into the cache or executed within the last *X* minutes (where *X* is a positive integer value). If the argument is not specified, all entries in the cache are returned.

- '`<update_boundary_time>evmon_name</update_boundary_time>`'

Updates the event monitor boundary timestamp to the current time for the package cache event monitor specified by *evmon_name*. If this event monitor specifies where *updated_since_boundary_time* as an output criteria in its WHERE clause, only package cache entries that subsequently have their metrics updated are captured when evicted from the package cache. This operation only has an effect if the specified package cache event monitor is active when the command is issued.

- '`<stmt_details>>true</stmt_details>`' or '`<stmt_details>>false</stmt_details>`'

Includes or excludes the *stmt_text* and *comp_env_desc* data in the resulting XML document. This allows you to exclude these relatively large portions of the document when you do not need them (for example, if you are using the XML document to provide input for the MON_FORMAT_XML_* table functions that return formatted row-based output). If this argument tag is not specified, the *stmt_text* and *comp_env_desc* data are included by default.

Each input argument can be specified only once. The search argument tags must be specified in lowercase.

member

An optional input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set.

Information returned

Table 143. Information returned for MON_GET_PKG_CACHE_STMT_DETAILS

| Column Name | Data Type | Description or corresponding monitor element |
|---------------|--------------------------|--|
| MEMBER | SMALLINT | member - Database member |
| SECTION_TYPE | CHAR(1) | section_type - Section type indicator. |
| EXECUTABLE_ID | VARCHAR(32) FOR BIT DATA | executable_id - Executable ID. |
| DETAILS | BLOB(20M) | XML document containing detailed metrics for the unit of work. See Table 144 for a description of the elements in this document. |

Table 144. Detailed metrics returned for MON_GET_PKG_CACHE_STMT_DETAILS

| Element Name | Data Type | Description |
|---------------------------------|-----------|---|
| active_col_vector_consumers_top | xs:long | active_col_vector_consumers_top - Active columnar_vector consumers high watermark |

Table 144. Detailed metrics returned for MON_GET_PKG_CACHE_STMT_DETAILS (continued)

| Element Name | Data Type | Description |
|-------------------------------|-----------------------|--|
| active_hash_grpbys_top | xs:long | active_hash_grpbys_top - Active hash Group By operations high watermark |
| active_hash_joins_top | xs:long | active_hash_joins_top - Active hash joins operations high watermark |
| active_olap_funcs_top | xs:long | active_olap_funcs_top - Active OLAP functions operations high watermark |
| active_peas_top | xs:long | active_peas_top - Active partial early aggregations operations high watermark |
| active_peds_top | xs:long | active_peds_top - Active partial early distinct operations high watermark |
| active_sort_consumers_top | xs:long | active_sort_consumers_top - Active sort consumers high watermark |
| active_sorts_top | xs:long | active_sorts_top - Active Sorts high watermark |
| adm_bypass_act_total | xs:long | Reserved for future use. |
| adm_overflows | xs:long | Reserved for future use. |
| audit_events_total | xs:long | audit_events_total - Total audit events |
| audit_file_write_wait_time | xs:long | audit_file_write_wait_time - Audit file write wait time |
| audit_file_writes_total | xs:long | audit_file_writes_total - Total Audit files written |
| audit_subsystem_wait_time | xs:long | audit_subsystem_wait_time - Audit subsystem wait time |
| audit_subsystem_waits_total | xs:long | audit_subsystem_waits_total - Total audit subsystem waits |
| cf_wait_time | xs:long | cf_wait_time - cluster caching facility wait time monitor element |
| cf_waits | xs:long | cf_waits - Number of cluster caching facility waits monitor element |
| col_synopsis_rows_inserted | xs:nonNegativeInteger | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| col_vector_consumer_overflows | xs:long | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |
| comm_exit_wait_time | xs:long | comm_exit_wait_time - Communication exit wait time monitor element |
| comm_exit_waits | xs:long | comm_exit_waits - Communication exit number of waits monitor element |
| comp_env_desc | xs:hexBinary(10240) | comp_env_desc - Compilation environment handle. You can use the existing COMPILATION_ENV table function to get the detailed compilation environment of the specific statement if needed. |
| coord_stmt_exec_time | xs:long | coord_stmt_exec_time - Execution time for statement by coordinator agent |
| deadlocks | xs:long | deadlocks - Deadlocks detected |
| diaglog_write_wait_time | xs:long | diaglog_write_wait_time - Diag log write time |
| diaglog_writes_total | xs:long | diaglog_writes_total - Diag log total writes |
| direct_read_reqs | xs:long | direct_read_reqs - Direct read requests |
| direct_read_time | xs:long | direct_read_time - Direct read time |
| direct_reads | xs:long | direct_reads - Direct reads from database |
| direct_write_reqs | xs:long | direct_write_reqs - Direct write requests |
| direct_write_time | xs:long | direct_write_time - Direct write time |
| direct_writes | xs:long | direct_writes - Direct writes to database |
| disabled_peds | xs:long | disabled_peds - Disabled partial early distincts |

Table 144. Detailed metrics returned for MON_GET_PKG_CACHE_STMT_DETAILS (continued)

| Element Name | Data Type | Description |
|------------------------------|-----------------------|--|
| effective_isolation | xs:string(2) | effective_isolation - Effective isolation. This is the isolation value in effect for the section; it can be different from what it was originally requested at compilation time. |
| estimated_runtime | xs:long | Reserved for future use. |
| estimated_sort_consumers_top | xs:long | Reserved for future use. |
| estimated_sort_shrheap_top | xs:long | Reserved for future use. |
| evmon_wait_time | xs:nonNegativeInteger | evmon_wait_time - Event monitor wait time |
| evmon_waits_total | xs:nonNegativeInteger | evmon_waits_total - Event monitor total waits |
| executable_id | xs:hexBinary(32) | executable_id - Executable ID. |
| ext_table_read_volume | xs:nonNegativeInteger | ext_table_read_volume - Total data read by external table readers monitor element |
| ext_table_rcv_volume | xs:nonNegativeInteger | ext_table_rcv_volume - Total data received from external table readers monitor element |
| ext_table_rcv_wait_time | xs:nonNegativeInteger | ext_table_rcv_wait_time - Total agent wait time for external table readers monitor element |
| ext_table_rcvs_total | xs:nonNegativeInteger | ext_table_rcvs_total - Total row batches received from external table readers monitor element |
| ext_table_send_volume | xs:nonNegativeInteger | ext_table_send_volume - Total data sent to external table writers monitor element |
| ext_table_send_wait_time | xs:nonNegativeInteger | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| ext_table_sends_total | xs:nonNegativeInteger | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| ext_table_write_volume | xs:nonNegativeInteger | ext_table_write_volume - Total data written by external table writers monitor element |
| fcm_message_rcv_volume | xs:long | fcm_message_rcv_volume - FCM message rcv volume |
| fcm_message_rcv_wait_time | xs:long | fcm_message_rcv_wait_time - FCM message rcv wait time |
| fcm_message_rcv_waits_total | xs:nonNegativeInteger | fcm_message_rcv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| fcm_message_rcvs_total | xs:long | fcm_message_rcvs_total - FCM message rcvs total |
| fcm_message_send_volume | xs:long | fcm_message_send_volume - FCM message send volume |
| fcm_message_send_wait_time | xs:long | fcm_message_send_wait_time - FCM message send wait time |
| fcm_message_send_waits_total | xs:nonNegativeInteger | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| fcm_message_sends_total | xs:long | fcm_message_sends_total - FCM message sends total |
| fcm_rcv_volume | xs:long | fcm_rcv_volume - FCM rcv volume |
| fcm_rcv_wait_time | xs:long | fcm_rcv_wait_time - FCM rcv wait time |
| fcm_rcvs_total | xs:long | fcm_rcvs_total - FCM rcvs total |
| fcm_send_volume | xs:long | fcm_send_volume - FCM send volume |
| fcm_send_wait_time | xs:long | fcm_send_wait_time - FCM send wait time |
| fcm_send_waits_total | xs:nonNegativeInteger | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| fcm_sends_total | xs:long | fcm_sends_total - FCM sends total |
| fcm_tq_rcv_volume | xs:long | fcm_tq_rcv_volume - FCM tablequeue rcv volume |
| fcm_tq_rcv_wait_time | xs:long | fcm_tq_rcv_wait_time - FCM tablequeue rcv wait time |
| fcm_tq_rcvs_total | xs:long | fcm_tq_rcvs_total - FCM tablequeue rcvs total |
| fcm_tq_send_volume | xs:long | fcm_tq_send_volume - FCM tablequeue send volume |
| fcm_tq_send_wait_time | xs:long | fcm_tq_send_wait_time - FCM tablequeue send wait time |
| fcm_tq_sends_total | xs:long | fcm_tq_sends_total - FCM tablequeue send total |

Table 144. Detailed metrics returned for MON_GET_PKG_CACHE_STMT_DETAILS (continued)

| Element Name | Data Type | Description |
|---------------------------|-----------------------|---|
| fed_rows_deleted | xs:nonNegativeInteger | fed_rows_deleted - Rows deleted by a federation system monitor element |
| fed_rows_inserted | xs:nonNegativeInteger | fed_rows_inserted - Rows inserted by a federation system monitor element |
| fed_rows_read | xs:nonNegativeInteger | fed_rows_read - Rows read by a federation system monitor element |
| fed_rows_updated | xs:nonNegativeInteger | fed_rows_updated - Rows updated by a federation system monitor element |
| fed_wait_time | xs:nonNegativeInteger | fed_wait_time - Time spent by a federation server monitor element |
| fed_waits_total | xs:nonNegativeInteger | fed_waits_total - Total number of execution times for a federation server monitor element |
| hash_grpby_overflows | xs:long | hash_grpby_overflows - Hash group by overflows |
| hash_join_overflows | xs:long | hash_join_overflows - Hash Join Overflows monitor element |
| hash_join_small_overflows | xs:long | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| ida_rcv_volume | xs:nonNegativeInteger | ida_rcv_volume - Total data volume received monitor element

The total volume of data the database server received from an in-database analytics process. The value is reported in bytes. |
| ida_rcv_wait_time | xs:nonNegativeInteger | ida_rcv_wait_time - Time spent waiting to receive data monitor element

The total amount of time spent waiting to receive data from an in-database analytics process. |
| ida_rcvs_total | xs:nonNegativeInteger | ida_rcvs_total - Number of times data received monitor element

The total number of times data was received from an in-database analytics process. |
| ida_send_volume | xs:nonNegativeInteger | ida_send_volume - Total data volume sent monitor element

The total volume of data sent from the database server to an in-database analytics process. The value is reported in bytes. |
| ida_send_wait_time | xs:nonNegativeInteger | ida_send_wait_time - Time spent waiting to send data monitor element

The total amount of time spent waiting to send data to an in-database analytics process. |
| ida_sends_total | xs:nonNegativeInteger | ida_sends_total - Number of times data sent monitor element

The total number of times data was sent to an in-database analytics process. |
| insert_timestamp | xs:dateTime | insert_timestamp - Statement insert timestamp |
| int_rows_deleted | xs:long | int_rows_deleted - Internal Rows Deleted monitor element |
| int_rows_inserted | xs:long | int_rows_inserted - Internal Rows Inserted monitor element |
| int_rows_updated | xs:long | int_rows_updated - Internal Rows Updated monitor element |
| last_metrics_update | xs:dateTime | last_metrics_update - Metrics last update timestamp |
| lob_prefetch_reqs | xs:nonNegativeInteger | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| lob_prefetch_wait_time | xs:nonNegativeInteger | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| lock_escals | xs:long | lock_escals - Number of lock escalations |

Table 144. Detailed metrics returned for MON_GET_PKG_CACHE_STMT_DETAILS (continued)

| Element Name | Data Type | Description |
|---------------------------------------|-----------------------|---|
| lock_escals_global | xs:long | lock_escals_global - Number of global lock escalations monitor element |
| lock_escals_locklist | xs:long | lock_escals_locklist - Number of locklist lock escalations monitor element |
| lock_escals_maxlocks | xs:long | lock_escals_maxlocks - Number of maxlocks lock escalations monitor element |
| lock_timeouts | xs:long | lock_timeouts - Number of lock timeouts |
| lock_timeouts_global | xs:long | lock_timeouts_global - Lock timeouts global monitor element |
| lock_wait_time | xs:long | lock_wait_time - Time waited on locks |
| lock_wait_time_global | xs:long | lock_wait_time_global - Lock wait time global monitor element |
| lock_waits | xs:long | lock_waits - Lock waits |
| lock_waits_global | xs:long | lock_waits_global - Lock waits global monitor element |
| log_buffer_wait_time | xs:long | log_buffer_wait_time - Log buffer wait time |
| log_disk_wait_time | xs:long | log_disk_wait_time - Log disk wait time |
| log_disk_waits_total | xs:long | log_disk_waits_total - Log disk waits total |
| max_coord_stmt_exec_time | xs:nonNegativeInteger | max_coord_stmt_exec_time - Maximum coordinator statement execution time |
| max_coord_stmt_exec_time_args | logical-grouping | max_coord_stmt_exec_time_args - Maximum coordinator statement execution time arguments |
| max_coord_stmt_exec_timestamp | xs:dateTime | max_coord_stmt_exec_timestamp - Maximum coordinator statement execution timestamp |
| member | xs:short | member - Database member |
| num_coord_exec | xs:long | num_coord_exec - Number of executions by coordinator agent |
| num_coord_exec_with_metrics | xs:long | num_coord_exec_with_metrics - Number of executions by coordinator agent with metrics |
| num_exec_with_metrics | xs:nonNegativeInteger | num_exec_with_metrics - Number of executions with metrics collected. |
| num_executions | xs:nonNegativeInteger | num_executions - Statement executions |
| num_log_buffer_full | xs:long | num_log_buffer_full - Number of full log buffers |
| num_lw_thresh_exceeded | xs:long | num_lw_thresh_exceeded - Number of thresholds exceeded |
| num_routines | xs:int | num_routines -Number of routines |
| num_working_copies | xs:long | Number of working copies. |
| olap_func_overflows | xs:long | olap_func_overflows - OLAP Function Overflows monitor element |
| package_name | xs:string(128) | package_name - Package name. This output is valid for static SQL statements only. A NULL value is returned if the statement is dynamic. |
| package_schema | xs:string(128) | package_schema - Package schema. This output is valid for static SQL statements only. A NULL value is returned if the statement is dynamic. |
| package_version_id | xs:string(64) | package_version_id - Package version. This output is valid for static SQL statements only. This element is not produced if the statement is dynamic or if you did not specify the package version for static statement. If you did not specify the package version identifier when the package was created, an empty string is returned for a static statement. |
| planid | xs:long | planid - Query plan ID monitor element |
| pool_col_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |

Table 144. Detailed metrics returned for MON_GET_PKG_CACHE_STMT_DETAILS (continued)

| Element Name | Data Type | Description |
|---|-----------------------|--|
| pool_col_gbp_invalid_pages | xs:nonNegativeInteger | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| pool_col_gbp_l_reads | xs:nonNegativeInteger | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| pool_col_gbp_p_reads | xs:nonNegativeInteger | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| pool_col_l_reads | xs:nonNegativeInteger | pool_col_l_reads - Buffer pool column-organized logical reads |
| pool_col_lbp_pages_found | xs:nonNegativeInteger | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found |
| pool_col_p_reads | xs:nonNegativeInteger | pool_col_p_reads - Buffer pool column-organized physical reads |
| pool_col_writes | xs:nonNegativeInteger | pool_col_writes - Buffer pool column-organized writes |
| pool_data_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| pool_data_gbp_invalid_pages | xs:long | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages |
| pool_data_gbp_l_reads | xs:long | pool_data_gbp_l_reads - Group buffer pool invalid data pages monitor element |
| pool_data_gbp_l_reads | xs:long | pool_data_gbp_l_reads - Group buffer pool data logical reads monitor element |
| pool_data_gbp_p_reads | xs:long | pool_data_gbp_p_reads - Group buffer pool data physical reads monitor element |
| pool_data_l_reads | xs:long | pool_data_l_reads - Buffer pool data logical reads |
| pool_data_lbp_pages_found | xs:long | pool_data_lbp_pages_found - Local buffer pool found data pages monitor element |
| pool_data_p_reads | xs:long | pool_data_p_reads - Buffer pool data physical reads |
| pool_data_writes | xs:long | pool_data_writes - Buffer pool data writes |
| pool_failed_async_col_reqs | xs:nonNegativeInteger | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| pool_failed_async_data_reqs | xs:nonNegativeInteger | pool_failed_async_data_reqs - Failed data prefetch requests monitor element |
| pool_failed_async_index_reqs | xs:nonNegativeInteger | pool_failed_async_index_reqs - Failed index prefetch requests monitor element |
| pool_failed_async_other_reqs | xs:nonNegativeInteger | pool_failed_async_other_reqs - Failed non-prefetch requests monitor element |
| pool_failed_async_temp_data_reqs | xs:nonNegativeInteger | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element |
| pool_failed_async_temp_index_reqs | xs:nonNegativeInteger | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element |
| pool_failed_async_temp_col_reqs | xs:nonNegativeInteger | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |
| pool_failed_async_temp_xda_reqs | xs:nonNegativeInteger | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element |
| pool_failed_async_xda_reqs | xs:nonNegativeInteger | pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element |
| pool_index_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| pool_index_gbp_invalid_pages | xs:long | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages monitor element |
| pool_index_gbp_l_reads | xs:long | pool_index_gbp_l_reads - Group buffer pool index logical reads monitor element |

Table 144. Detailed metrics returned for MON_GET_PKG_CACHE_STMT_DETAILS (continued)

| Element Name | Data Type | Description |
|------------------------------------|-----------------------|--|
| pool_index_gbp_p_reads | xs:long | pool_index_gbp_p_reads - Group buffer pool index physical reads monitor elements |
| pool_index_l_reads | xs:long | pool_index_l_reads - Buffer pool index logical reads |
| pool_index_lbp_pages_found | xs:long | pool_index_lbp_pages_found - Local buffer pool index pages found monitor element |
| pool_index_p_reads | xs:long | pool_index_p_reads - Buffer pool index physical reads |
| pool_index_writes | xs:long | pool_index_writes - Buffer pool index writes |
| pool_queued_async_col_reqs | xs:nonNegativeInteger | pool_queued_async_col_reqs - Column-organized prefetch requests |
| pool_queued_async_data_pages | xs:nonNegativeInteger | pool_queued_async_data_pages - Data pages prefetch requests monitor element |
| pool_queued_async_data_reqs | xs:nonNegativeInteger | pool_queued_async_data_reqs - Data prefetch requests monitor element |
| pool_queued_async_index_reqs | xs:nonNegativeInteger | pool_queued_async_index_reqs - Index prefetch requests monitor element |
| pool_queued_async_index_pages | xs:nonNegativeInteger | pool_queued_async_index_pages - Index pages prefetch requests monitor element |
| pool_queued_async_col_pages | xs:nonNegativeInteger | pool_queued_async_col_pages - Column-organized page prefetch requests |
| pool_queued_async_temp_col_pages | xs:nonNegativeInteger | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| pool_queued_async_temp_col_reqs | xs:nonNegativeInteger | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| pool_queued_async_other_reqs | xs:nonNegativeInteger | pool_queued_async_other_reqs - Non-prefetch requests monitor element |
| pool_queued_async_temp_data_pages | xs:nonNegativeInteger | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_data_reqs | xs:nonNegativeInteger | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_index_pages | xs:nonNegativeInteger | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_index_reqs | xs:nonNegativeInteger | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_xda_pages | xs:nonNegativeInteger | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_xda_reqs | xs:nonNegativeInteger | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element |
| pool_queued_async_xda_pages | xs:long | pool_queued_async_xda_pages - XDA pages prefetch requests monitor element |
| pool_queued_async_xda_reqs | xs:nonNegativeInteger | pool_queued_async_xda_reqs - XDA prefetch requests monitor element |
| pool_read_time | xs:long | pool_read_time - Total buffer pool physical read time |
| pool_temp_col_l_reads | xs:nonNegativeInteger | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| pool_temp_col_p_reads | xs:nonNegativeInteger | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| pool_temp_data_l_reads | xs:long | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| pool_temp_data_p_reads | xs:long | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| pool_temp_index_l_reads | xs:long | pool_temp_index_l_reads - Buffer pool temporary index logical reads |

Table 144. Detailed metrics returned for MON_GET_PKG_CACHE_STMT_DETAILS (continued)

| Element Name | Data Type | Description |
|---------------------------------------|-----------------------|--|
| pool_temp_index_p_reads | xs:long | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| pool_temp_xda_l_reads | xs:long | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads monitor element |
| pool_temp_xda_p_reads | xs:long | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| pool_write_time | xs:long | pool_write_time - Total buffer pool physical write time |
| pool_xda_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |
| pool_xda_gbp_invalid_pages | xs:nonNegativeInteger | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| pool_xda_gbp_l_reads | xs:nonNegativeInteger | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |
| pool_xda_gbp_p_reads | xs:nonNegativeInteger | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |
| pool_xda_l_reads | xs:long | pool_xda_l_reads - Buffer pool XDA data logical reads monitor element |
| pool_xda_l_reads | xs:long | pool_xda_l_reads - Buffer pool XDA data logical reads |
| pool_xda_lbp_pages_found | xs:nonNegativeInteger | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |
| pool_xda_p_reads | xs:long | pool_xda_p_reads - Buffer pool XDA data physical reads |
| pool_xda_writes | xs:long | pool_xda_writes - Buffer pool XDA data writes |
| post_shrthreshold_hash_joins | xs:long | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| post_shrthreshold_sorts | xs:long | post_shrthreshold_sorts - Post shared threshold sorts |
| post_threshold_col_vector_consumers | xs:long | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| post_threshold_hash_grpbys | xs:long | post_threshold_hash_grpbys - Hash group by threshold
post_threshold_hash_grpbys - Hash group by threshold |
| post_threshold_hash_joins | xs:long | post_threshold_hash_joins - Hash Join Threshold monitor element |
| post_threshold_olap_funcs | xs:long | post_threshold_olap_funcs - OLAP function threshold monitor element |
| post_threshold_peas | xs:long | post_threshold_peas - Partial early aggregation threshold |
| post_threshold_peds | xs:long | post_threshold_peds - Partial early distincts threshold |
| post_threshold_sorts | xs:long | post_threshold_sorts - Post threshold sorts |
| prefetch_wait_time | xs:nonNegativeInteger | prefetch_wait_time - Time waited for prefetch |
| prefetch_waits | xs:nonNegativeInteger | prefetch_waits - Prefetcher wait count monitor element |
| prep_time | xs:nonNegativeInteger | prep_time - Preparation time Note that PREP_TIME is only valid for dynamic SQL statements. PREP_TIME is reported as 0 for static SQL statements. |
| prep_warning | xs:int | prep_warning - Prepare warning SQLCODE monitor element |
| prep_warning_reason | xs:int | prep_warning_reason - Prepare warning SQLCODE reason identifier monitor element |
| fcm_tq_send_waits_total | xs:nonNegativeInteger | fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |
| query_cost_estimate | xs:long | query_cost_estimate - Query cost estimate |
| query_data_tag_list | xs:string(32) | query_data_tag_list - Query data tag list |
| reclaim_wait_time | xs:long | reclaim_wait_time - Reclaim wait time monitor element |

Table 144. Detailed metrics returned for MON_GET_PKG_CACHE_STMT_DETAILS (continued)

| Element Name | Data Type | Description |
|--------------------------------|-----------------------|--|
| routine_id | xs:long | routine_id - Routine ID. For CALL statements provides the routine identifier for the target procedure. For all other types of statements the value is 0. |
| rows_deleted | xs:long | rows_deleted - Rows deleted monitor element |
| rows_inserted | xs:long | rows_inserted - Rows inserted monitor element |
| rows_modified | xs:long | rows_modified - Rows modified |
| rows_read | xs:long | rows_read - Rows read |
| rows_returned | xs:long | rows_returned - Rows returned |
| rows_updated | xs:long | rows_updated - Rows updated monitor element |
| section_number | xs:short | section_number - Section number. This element is not produced if the statement is dynamic. |
| section_type | xs:string(1) | section_type - Section type indicator. |
| semantic_env_id | xs:long | semantic_env_id - Query semantic compilation environment ID monitor element |
| sort_consumer_heap_top | xs:long | sort_consumer_heap_top - Individual private sort heap consumer high watermark |
| sort_consumer_shrheap_top | xs:long | sort_consumer_shrheap_top - Individual shared sort heap consumer high watermark |
| sort_heap_top | xs:long | sort_heap_top - Sort private heap high watermark |
| sort_overflows | xs:long | sort_overflows - Sort overflows |
| sort_shrheap_top | xs:long | sort_shrheap_top - Sort share heap high watermark |
| spacemappage_reclaim_wait_time | xs:long | spacemappage_reclaim_wait_time - Space map page reclaim wait time monitor element |
| stmt_exec_time | xs:long | stmt_exec_time - Statement execution time |
| stmt_pkg_cache_id | xs:long | stmt_pkgcache_id - Statement package cache identifier |
| stmt_comments | stmt_comments_type | stmt_comments - Comments in SQL statement text monitor element |
| stmt_text | xs:string(2097152) | stmt_text - SQL statement text |
| stmt_type_id | xs:string | stmt_type_id - Statement type identifier |
| stmt_value_data | xs:string(32768) | stmt_value_data - Value data |
| stmt_value_index | xs:nonNegativeInteger | stmt_value_index - Value index |
| stmt_value_isnull | xs:string(20) | stmt_value_isnull - Value has null value |
| stmt_value_isreopt | xs:string(20) | stmt_value_isreopt - Variable used for statement reoptimization |
| stmt_value_type | xs:string(255) | stmt_value_type - Value type |
| stmtid | xs:long | stmtid - Query statement ID monitor element |
| stmtno | xs:int | stmtno - Statement number monitor element |
| thresh_violations | xs:long | thresh_violations - Number of threshold violations |
| total_act_time | xs:long | total_act_time - Total activity time |
| total_act_wait_time | xs:long | total_act_wait_time - Total activity wait time |
| total_app_section_executions | xs:long | total_app_section_executions - Total section executions |
| total_col_executions | xs:nonNegativeInteger | total_col_executions - Total column-organized executions |
| total_col_proc_time | xs:nonNegativeInteger | total_col_proc_time - Total column-organized processing time |
| total_col_synopsis_executions | xs:nonNegativeInteger | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |
| total_col_synopsis_proc_time | xs:nonNegativeInteger | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| total_col_synopsis_time | xs:nonNegativeInteger | total_col_synopsis_time - Total column-organized synopsis table time monitor element |

Table 144. Detailed metrics returned for MON_GET_PKG_CACHE_STMT_DETAILS (continued)

| Element Name | Data Type | Description |
|-----------------------------------|-----------------------|--|
| total_col_time | xs:nonNegativeInteger | total_col_time - Total column-organized time |
| total_col_vector_consumers | xs:long | total_col_vector_consumers - Total columnar_vector consumers |
| total_cpu_time | xs:long | total_cpu_time - Total CPU time |
| total_disp_run_queue_time | xs:long | total_disp_run_queue_time - Total dispatcher run queue time |
| total_extended_latch_wait_time | xs:nonNegativeInteger | total_extended_latch_wait_time - Total extended latch wait time |
| total_extended_latch_waits | xs:nonNegativeInteger | total_extended_latch_waits - Total extended latch waits |
| total_hash_grpbys | xs:long | total_hash_grpbys - Total hash group by operations |
| total_hash_joins | xs:long | total_hash_joins - Total Hash Joins monitor element |
| total_hash_loops | xs:long | total_hash_loops - Total Hash Loops monitor element |
| total_index_build_proc_time | xs:long | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation |
| total_index_build_time | xs:long | total_index_build_time - Total time spent building indexes due to index creation |
| total_indexes_built | xs:long | total_indexes_built - Total number of indexes built |
| total_olap_funcs | xs:long | total_olap_funcs - Total OLAP Functions monitor element |
| total_peas | xs:long | total_peas - Total partial early aggregations |
| total_peds | xs:long | total_peds - Total partial early distincts |
| total_routine_invocations | xs:long | total_routine_invocations - Total routine invocations |
| total_routine_non_sect_proc_time | xs:long | total_routine_non_sect_proc_time - Non-section processing time |
| total_routine_non_sect_time | xs:long | total_routine_non_sect_time - Non-section routine execution time |
| total_routine_time | xs:long | total_routine_time - Total routine time |
| total_routine_user_code_time | xs:long | total_routine_user_code_time - Total routine user code time |
| total_routine_user_code_proc_time | xs:long | total_routine_user_code_proc_time - Total routine user code processing time |
| total_section_proc_time | xs:long | total_section_proc_time - Total section processing time |
| total_section_sort_proc_time | xs:long | total_section_sort_proc_time - Total section sort processing time |
| total_section_sort_time | xs:long | total_section_sort_time - Total section sort time |
| total_section_sorts | xs:long | total_section_sorts - Total section sorts |
| total_section_time | xs:long | total_section_time - Total section time |
| total_sorts | xs:long | total_sorts - Total Sorts |
| total_stats_fabrication_time | xs:nonNegativeInteger | total_stats_fabrication_time - Total statistics fabrication time |
| total_stats_fabrications | xs:nonNegativeInteger | total_stats_fabrications - Total statistics fabrications |
| total_sync_runstats | xs:nonNegativeInteger | total_sync_runstats - Total synchronous RUNSTATS activities |
| total_sync_runstats_time | xs:nonNegativeInteger | total_sync_runstats_time - Total synchronous RUNSTATS time |
| tq_sort_heap_rejections | xs:long | tq_sort_heap_rejections - Table queue sort heap rejections |
| tq_sort_heap_requests | xs:long | tq_sort_heap_requests - Table queue sort heap requests |
| tq_tot_send_spills | xs:long | tq_tot_send_spills - Total number of table queue buffers overflowed |
| valid | xs:string(1) | valid - Section validity indicator |
| wlm_queue_assignments_total | xs:long | wlm_queue_assignments_total - Workload manager total queue assignments |
| wlm_queue_time_total | xs:long | wlm_queue_time_total - Workload manager total queue time |

Usage notes

The metrics returned by this function represent the accumulation of all metrics for statements in the package cache. Statement metrics are rolled up to the package cache upon activity completion.

The schema for the XML document that is returned in the DETAILS column is available in the file `sqllib/misc/DB2MonRoutines.xsd`. Further details can be found in the file `sqllib/misc/DB2MonCommon.xsd`.

Examples

The first example demonstrates how to examine the package cache and select the 10 statements that have read and returned the largest number of rows. Additionally, the results show the cumulative amount of time spent executing each of these statements (in the STMT_EXEC_TIME output column).

```
SELECT SUBSTR(DETMETRICS.STMT_TEXT, 1, 40) STMT_TEXT,
       DETMETRICS.ROWS_RETURNED,
       DETMETRICS.STMT_EXEC_TIME
FROM TABLE(MON_GET_PKG_CACHE_STMT_DETAILS(CAST(NULL AS CHAR(1)),
      CAST(NULL AS VARCHAR(32) FOR BIT DATA),
      CAST(NULL AS CLOB(1K)), -1)) AS STMT_METRICS,
      XMLTABLE (XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'),
      '$DETMETRICS/db2_pkg_cache_stmt_details' PASSING
      XMLPARSE(DOCUMENT STMT_METRICS.DETAILS) as "DETMETRICS"
      COLUMNS "STMT_TEXT" CLOB PATH 'stmt_text',
      "ROWS_RETURNED" BIGINT PATH 'activity_metrics/rows_returned',
      "STMT_EXEC_TIME" BIGINT PATH 'activity_metrics/stmt_exec_time'
      ) AS DETMETRICS
ORDER BY rows_returned DESC
FETCH FIRST 10 ROWS ONLY
```

The following is an example of output from this query.

| STMT_TEXT | ROWS_RETURNED | STMT_EXEC_TIME |
|--|---------------|----------------|
| SELECT CREATOR, NAME, CTIME FROM SYSIBM. | 134 | 38 |
| SELECT SUBSTR(DETMETRICS.STMT_TEXT, 1, 4 | 44 | 336 |
| SELECT SUBSTR(DETMETRICS.STMT_TEXT, 1, 4 | 10 | 333 |
| SELECT COLNAME, TYPENAME FROM SYSCAT.CO | 10 | 6 |
| SELECT SUBSTR(DETMETRICS.STMT_TEXT, 1, 4 | 10 | 334 |
| SELECT TRIGNAME FROM SYSCAT.TRIGGERS WH | 8 | 1 |
| SELECT COUNT(*) FROM SYSCAT.TABLESPACES | 2 | 0 |
| SELECT POLICY FROM SYSTOOLS.POLICY WHERE | 1 | 0 |
| CALL SYSPROC.POLICY_INSTALL ('I','DB2Tab | 1 | 62 |
| CALL SYSPROC.POLICY_INSTALL ('I','DB2Tab | 1 | 64 |

10 record(s) selected.

The second example shows, for dynamic SQL statements that have waited on a lock while executing, the number of executions, number of lock waits and average time spent per lock wait. The output shows values accumulated over the lifetime of the package cache entries, but restricts information to statements that have executed within the last minute (by setting the modified_within argument tag to 1). The query excludes the statement details (*stmt_text* and *comp_env_desc* data) because they are not required and they are computationally expensive to report (by setting the stmt_details argument tag to false).

```
SELECT NUM_EXEC_WITH_METRICS, LOCK_WAITS,
       (LOCK_WAIT_TIME / LOCK_WAITS) AVG_LOCK_WAIT_TIME
FROM TABLE(MON_GET_PKG_CACHE_STMT_DETAILS('D', CAST(NULL
      AS VARCHAR(32) FOR BIT DATA),
      CLOB(
      '<modified_within>1</modified_within><stmt_details>false</stmt_details>'
      , -1))
```

```

AS STMT_METRICS,
XMLTABLE (XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'),
'$DETMETRICS/db2_pkg_cache_stmt_details' PASSING
XMLPARSE(DOCUMENT STMT_METRICS.DETAILS) as "DETMETRICS"
COLUMNS "NUM_EXEC_WITH_METRICS" BIGINT PATH 'num_exec_with_metrics',
"LOCK_WAITS" BIGINT PATH 'lock_waits',
"LOCK_WAIT_TIME" BIGINT PATH 'activity_metrics/lock_wait_time'
) AS DETMETRICS
WHERE LOCK_WAITS <> 0
ORDER BY AVG_LOCK_WAIT_TIME DESC

```

The following is an example of output from this query.

| NUM_EXEC_WITH_METRICS | LOCK_WAITS | AVG_LOCK_WAIT_TIME |
|-----------------------|------------|--------------------|
| 4 | 2 | 139 |
| 9 | 3 | 90 |

MON_GET_QUEUE_STATS table function - Return threshold queue statistics

The MON_GET_QUEUE_STATS function returns, for all active members, basic statistics for one or all threshold queues. This function returns one row of statistics for each queue.

Authorization

At least one of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None.

Syntax

```

▶▶ MON_GET_QUEUE_STATS ( ( predicate , [ domain ] ,
[ threshold_name ] , [ threshold_id ] )

```

The schema is SYSPROC.

Routine parameters

predicate

An input argument of type VARCHAR(27) that specifies for which queue or queues statistics are to be retrieved:

CONCDBC

Retrieve statistics for the **concurrent database coordinator activities** threshold queue.

DBCONN

Retrieve statistics for the **total database member connections** threshold queue.

SCCONN

Retrieve statistics for the **total service class member connections** threshold queue.

If this argument is null or an empty string, statistics are retrieved for all queues that meet the other criteria.

The specified value corresponds to the value in the THRESHOLDPREDICATE column of the SYSCAT.THRESHOLDS view.

domain

An input argument of type VARCHAR(18) that specifies a threshold domain:

DB Database.

SB Service subclass.

SP Service superclass.

WA Work action set.

If this argument is null or an empty string, statistics are retrieved for all queues that meet the other criteria.

The specified value corresponds to the value in the DOMAIN column of the SYSCAT.THRESHOLDS view.

threshold_name

An input argument of type VARCHAR(128) that specifies the name of a particular threshold for which statistics are to be retrieved. The specified name corresponds to the value in the THRESHOLDNAME column of the SYSCAT.THRESHOLDS view. If this argument is null or an empty string, statistics are retrieved for all thresholds that satisfy the other criteria.

threshold_id

An input argument of type INTEGER that specifies the ID of a particular threshold for which statistics are to be retrieved. The specified ID corresponds to the value in the THRESHOLDID column of the SYSCAT.THRESHOLDS view. If the argument is null or -1, statistics are retrieved for all thresholds that satisfy the other criteria.

Information returned

Table 145. Information returned by the MON_GET_QUEUE_STATS routine

| Column name | Data type | Description |
|---------------------|--------------|--|
| THRESHOLD_PREDICATE | VARCHAR(27) | threshold_predicate - Threshold predicate monitor element |
| THRESHOLD_DOMAIN | VARCHAR(18) | threshold_domain - Threshold domain monitor element |
| THRESHOLD_NAME | VARCHAR(128) | threshold_name - Threshold name monitor element |
| THRESHOLD_ID | INTEGER | thresholdid - Threshold ID monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

Table 145. Information returned by the MON_GET_QUEUE_STATS routine (continued)

| Column name | Data type | Description |
|-------------------------------|--------------|--|
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name monitor element |
| SERVICE_SUBCLASS_NAME | VARCHAR(128) | service_subclass_name - Service subclass name monitor element |
| WORK_ACTION_SET_NAME | VARCHAR(128) | work_action_set_name - Work action set name monitor element |
| WORK_CLASS_NAME | VARCHAR(128) | work_class_name - Work class name monitor element |
| WORKLOAD_NAME | VARCHAR(128) | workload_name - Workload name monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |
| QUEUE_SIZE_TOP | INTEGER | queue_size_top - Queue size top monitor element |
| QUEUE_TIME_TOTAL | BIGINT | queue_time_total - Queue time total monitor element |
| QUEUE_ASSIGNMENTS_TOTAL | BIGINT | queue_assignments_total - Queue assignments total monitor element |
| QUEUE_SIZE_CURRENT | INTEGER | Number of connections or activities in the queue. |
| QUEUE_TIME_LATEST | BIGINT | Time spent in the queue by the last connection or activity to leave the queue. Units are milliseconds. |
| QUEUE_EXIT_TIME_LATEST | TIMESTAMP | Time that the last connection or activity left the queue. |
| THRESHOLD_CURRENT_CONCURRENCY | INTEGER | Number of connections or activities that are currently running according to the threshold. |
| THRESHOLD_MAX_CONCURRENCY | INTEGER | Maximum number of connections or activities that the threshold allows to be concurrently running. |
| MEMBER | SMALLINT | member - Database member monitor element |

Usage notes

The function does not aggregate data across members (for one or more queues). However, you can use SQL queries to aggregate data, as shown in the example.

Examples

- The following query displays the basic statistics for all the queues on a system, across all members:

```
SELECT substr(THRESHOLD_NAME, 1, 6) THRESHNAME,
       THRESHOLD_PREDICATE,
       THRESHOLD_DOMAIN,
       MEMBER MEMB,
       QUEUE_SIZE_TOP,
       QUEUE_TIME_TOTAL,
       QUEUE_ASSIGNMENTS_TOTAL QUEUE_ASSIGN
FROM table(MON_GET_QUEUE_STATS('', '', '', -1)) as QSTATS
```

Sample output:

| THRESHNAME | THRESHOLD_PREDICATE | THRESHOLD_DOMAIN | MEMB | QUEUE_SIZE_TOP | QUEUE_TIME_TOTAL | QUEUE_ASSIGN |
|------------|---------------------|------------------|------|----------------|------------------|--------------|
| LIMIT1 | CONCDBC | DB | 0 | 12 | 1238540 | 734 |
| LIMIT2 | SCCONN | SP | 0 | 4 | 741249 | 24 |
| LIMIT3 | DBCONN | DB | 0 | 7 | 412785 | 128 |

MON_GET_REBALANCE_STATUS table function - get rebalance progress for a table space

The MON_GET_REBALANCE_STATUS table function returns the status of a rebalance operation on a table space.

Syntax

►►—MON_GET_REBALANCE_STATUS—(—*tblsp_name*—,—*dbpartitionnum*—)——————►◄

The schema is SYSPROC.

Table function parameters

tblsp_name

An input argument of type VARCHAR(128) that specifies the name of the table space to query. If the argument value is null, the function returns information for all table spaces.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition in the same instance as the currently connected database when calling this function. Specify -1 for the current database partition, or -2 for all active database partitions. If the NULL value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

List all active rebalance operations on the database and display their current state and progress.

```
select
    varchar(tblsp_name, 30) as tblsp_name,
    dbpartitionnum,
    member,
    rebalancer_mode,
    rebalancer_status,
    rebalancer_extents_remaining,
    rebalancer_extents_processed,
    rebalancer_start_time
from table(mon_get_rebalance_status(NULL,-2)) as t
```

| TBSP_NAME | DBPARTITIONNUM | MEMBER | REBALANCER_MODE |
|-----------|----------------|--------|-----------------|
| ----- | ----- | ----- | ----- |

SYSCATSPACE

0

0 REV_REBAL

```

REBALANCER_STATUS REBALANCER_EXTENTS_REMAINING REBALANCER_EXTENTS_PROCESSED
-----
ACTIVE                                     6517                                     4

```

```

REBALANCER_START_TIME
-----
2011-12-01-12.08.16.000000

```

1 record(s) selected.

Usage notes

The MON_GET_REBALANCE_STATUS table function only returns data for a table space if a rebalance is in progress. Otherwise, no data is returned.

Information returned

Table 146. Information returned for MON_GET_REBALANCE_STATUS

| Column Name | Data Type | Description or corresponding monitor element |
|--------------------------------------|--------------|--|
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name monitor element |
| TBSP_ID | BIGINT | tablespace_id - Table space identification monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number. Indicates the data being rebalanced. |
| MEMBER | SMALLINT | member - Database member. Indicates the processing member doing the rebalance. Rebalance is not supported on Db2 pureScale |
| REBALANCER_MODE | VARCHAR(30) | rebalancer_mode - Rebalancer mode monitor element |
| REBALANCER_STATUS | VARCHAR(10) | rebalancer_status - Rebalancer status monitor element
rebalancer_status - Rebalancer status monitor element |
| REBALANCER_EXTENTS_REMAINING | BIGINT | rebalancer_extents_remaining - Total number of extents to be processed by the rebalancer |
| REBALANCER_EXTENTS_PROCESSED | BIGINT | rebalancer_extents_processed - Number of extents the rebalancer has processed |
| REBALANCER_PRIORITY | BIGINT | rebalancer_priority - Current rebalancer priority |
| REBALANCER_LAST_EXTENT_MOVED | BIGINT | rebalancer_last_extent_moved - Last extent moved by the rebalancer |
| REBALANCER_START_TIME | TIMESTAMP | rebalancer_start_time - Rebalancer start time |
| REBALANCER_RESTART_TIME | TIMESTAMP | rebalancer_restart_time - Rebalancer restart time |
| REBALANCER_SOURCE_STORAGE_GROUP_NAME | VARCHAR(128) | rebalancer_source_storage_group_name - Rebalancer source storage group name |
| REBALANCER_SOURCE_STORAGE_GROUP_ID | INTEGER | rebalancer_source_storage_group_id - Rebalancer source storage group identifier |
| REBALANCER_TARGET_STORAGE_GROUP_NAME | VARCHAR(128) | rebalancer_target_storage_group_name - Rebalancer target storage group name |
| REBALANCER_TARGET_STORAGE_GROUP_ID | INTEGER | rebalancer_target_storage_group_id - Rebalancer target storage group identifier |

MON_GET_ROUTINE table function - get aggregated routine execution metrics

The MON_GET_ROUTINE table function returns aggregated execution metrics for procedures, external procedures, compiled functions, external functions, compiled triggers, and anonymous blocks invoked since the database was activated.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- SQLADM authority
- DBADM authority

Default PUBLIC privilege

None

Syntax

```
►► MON_GET_ROUTINE (—routine_type—, —routines_chema—, —————►  
► —routine_module—, —routine_name—, —member—) —————►◄
```

The schema is SYSPROC.

Routine parameters

routine_type

An input parameter of type CHAR(2) that specifies the type of routine or compiled trigger for which data is returned:

- "P" for a procedure
- "SP" for the specific name of a procedure
- "F" for a compiled function
- "SF" for the specific name of a compiled function
- "T" for a compiled trigger
- "C" for a dynamically prepared compound SQL statement or an anonymous block in PL/SQL

If the "P", "F", "T" or "C" routine type is specified and no input name is specified, all routines of the given type will be returned. If the routine type is "SP" or "SF", the specific name of the routine must be provided.

If the routine type is an empty string, NULL, or blanks, all routines of all types are returned.

routine_schema

An input parameter of type VARCHAR(128) that specifies the schema for the routine or trigger. For dynamically prepared compound SQL statements or anonymous blocks, the schema can be determined using the MON_GET_SECTION_ROUTINES table function. Use NULL or an empty string to return the routines and triggers in all schemas. This parameter is case sensitive.

routine_module_name

An input parameter of type VARCHAR(128) that specifies the name of the module for the input routine, if applicable. Use NULL or an empty string to return the routines in all modules. This parameter is case sensitive.

routine_name

An input parameter of type VARCHAR(128) that specifies the name of the routine. If the input parameter is "SP" or "SF", the specific name of the routine must be provided. For dynamically prepared compound SQL statements or anonymous blocks, the name can be determined using the MON_GET_SECTION_ROUTINES table function. Use NULL or an empty string to return all routines that match the other input parameters. This parameter is case sensitive.

member

An input parameter of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is implicitly set.

Information returned

Table 147. Information returned for MON_GET_ROUTINE

| Column name | Data type | Description or corresponding monitor element |
|-----------------------------|--------------------------|---|
| ROUTINE_TYPE | CHAR(1) | routine_type - Routine type monitor element |
| ROUTINE_SCHEMA | VARCHAR(128) | routine_schema - Routine schema monitor element |
| ROUTINE_MODULE_NAME | VARCHAR(128) | routine_module_name - Routine module name monitor element |
| ROUTINE_NAME | VARCHAR(128) | routine_name - Routine name monitor element |
| SPECIFIC_NAME | VARCHAR(128) | specific_name - Specific name monitor element |
| DYN_COMPOUND_EXEC_ID | VARCHAR(32) FOR BIT DATA | dyn_compound_exec_id - Dynamic compound statement executable identifier monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| ROUTINE_ID | INTEGER | routine_id - Routine ID monitor element |
| SUBROUTINE_ID | INTEGER | subroutine_id - Subroutine identifier monitor element |
| LIB_ID | BIGINT | lib_id - Library identifier monitor element |
| EXEC_LIST_MEM_EXCEEDED | CHAR(1) | exec_list_mem_exceeded - Execution list memory exceeded monitor element |
| EXEC_LIST_CLEANUP_TIME | TIMESTAMP | exec_list_cleanup_time - Execution list cleanup time monitor element |
| TOTAL_TIMES_ROUTINE_INVOKED | BIGINT | total_times_routine_invoked - Total routine invoked occurrences monitor element |
| TOTAL_NESTED_INVOCATIONS | BIGINT | total_nested_invocations - Total nested invocations monitor element |
| TOTAL_ROUTINE_COORD_TIME | BIGINT | total_routine_coord_time - Total routine coordinator time monitor element |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| ACT_ABORTED_TOTAL | BIGINT | act_aborted_total - Total aborted activities |
| ACT_COMPLETED_TOTAL | BIGINT | act_completed_total - Total completed activities |
| ACT_REJECTED_TOTAL | BIGINT | act_rejected_total - Total rejected activities |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer pool XDA data physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer pool XDA data writes |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| FCM_RECV_VOLUME | BIGINT | fcm_recv_volume - FCM received volume |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|-----------|---|
| FCM_RECVS_TOTAL | BIGINT | fcm_recvs_total - FCM receives total |
| FCM_SEND_VOLUME | BIGINT | fcm_send_volume - FCM send volume |
| FCM_SENDS_TOTAL | BIGINT | fcm_sends_total - FCM sends total |
| FCM_RECV_WAIT_TIME | BIGINT | fcm_rcv_wait_time - FCM received wait time |
| FCM_SEND_WAIT_TIME | BIGINT | fcm_send_wait_time - FCM send wait time |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOG_BUFFER_WAIT_TIME | BIGINT | log_buffer_wait_time - Log buffer wait time |
| NUM_LOG_BUFFER_FULL | BIGINT | num_log_buffer_full - Number of times full log buffer caused agents to wait |
| LOG_DISK_WAIT_TIME | BIGINT | log_disk_wait_time - Log disk wait time |
| LOG_DISK_WAITS_TOTAL | BIGINT | log_disk_waits_total - Total log disk waits |
| RQSTS_COMPLETED_TOTAL | BIGINT | rqsts_completed_total - Total requests completed |
| ROWS_MODIFIED | BIGINT | rows_modified - Rows modified |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_RETURNED | BIGINT | rows_returned - Rows returned |
| TOTAL_APP_RQST_TIME | BIGINT | total_app_rqst_time - Total application request time |
| TOTAL_RQST_TIME | BIGINT | total_rqst_time - Total request time. |
| WLM_QUEUE_TIME_TOTAL | BIGINT | wlm_queue_time_total - Workload manager total queue time |
| WLM_QUEUE_ASSIGNMENTS_TOTAL | BIGINT | wlm_queue_assignments_total - Workload manager total queue assignments |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time. |
| TOTAL_WAIT_TIME | BIGINT | total_wait_time - Total wait time |
| APP_RQSTS_COMPLETED_TOTAL | BIGINT | app_rqsts_completed_total - Total application requests completed |
| TOTAL_SECTION_SORT_TIME | BIGINT | total_section_sort_time - Total section sort time |
| TOTAL_SECTION_SORT_PROC_TIME | BIGINT | total_section_sort_proc_time - Total section sort processing time |
| TOTAL_SECTION_SORTS | BIGINT | total_section_sorts - Total section sorts |
| TOTAL_SORTS | BIGINT | total_sorts - Total sorts |
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post threshold sorts |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post shared threshold sorts |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| TOTAL_COMPILE_TIME | BIGINT | total_compile_time - Total compile time |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------------|-----------|---|
| TOTAL_COMPILE_PROC_TIME | BIGINT | total_compile_proc_time - Total compile processing time |
| TOTAL_COMPILATIONS | BIGINT | total_compilations - Total compilations |
| TOTAL_IMPLICIT_COMPILE_TIME | BIGINT | total_implicit_compile_time - Total implicit compile time |
| TOTAL_IMPLICIT_COMPILE_PROC_TIME | BIGINT | total_implicit_compile_proc_time - Total implicit compile processing time |
| TOTAL_IMPLICIT_COMPILATIONS | BIGINT | total_implicit_compilations - Total implicit complications |
| TOTAL_SECTION_TIME | BIGINT | total_section_time - Total section time |
| TOTAL_SECTION_PROC_TIME | BIGINT | total_section_proc_time - Total section processing time |
| TOTAL_APP_SECTION_EXECUTIONS | BIGINT | total_app_section_executions - Total application section executions |
| TOTAL_ACT_TIME | BIGINT | total_act_time - Total activity time |
| TOTAL_ACT_WAIT_TIME | BIGINT | total_act_wait_time - Total activity wait time |
| ACT_RQSTS_TOTAL | BIGINT | act_rqsts_total - Total activity requests |
| TOTAL_ROUTINE_TIME | BIGINT | total_routine_time - Total routine time. |
| TOTAL_ROUTINE_INVOCATIONS | BIGINT | total_routine_invocations - Total routine invocations. The total number of times a routine was invoked by this routine. |
| TOTAL_COMMIT_TIME | BIGINT | total_commit_time - Total commit time |
| TOTAL_COMMIT_PROC_TIME | BIGINT | total_commit_proc_time - Total commits processing time |
| TOTAL_APP_COMMITS | BIGINT | total_app_commits - Total application commits |
| INT_COMMITS | BIGINT | int_commits - Internal commits |
| TOTAL_ROLLBACK_TIME | BIGINT | total_rollback_time - Total rollback time |
| TOTAL_ROLLBACK_PROC_TIME | BIGINT | total_rollback_proc_time - Total rollback processing time |
| TOTAL_APP_ROLLBACKS | BIGINT | total_app_rollbacks - Total application rollbacks |
| INT_ROLLBACKS | BIGINT | int_rollbacks - Internal rollbacks |
| TOTAL_RUNSTATS_TIME | BIGINT | total_runstats_time - Total runtime statistics |
| TOTAL_RUNSTATS_PROC_TIME | BIGINT | total_runstats_proc_time - Total runtime statistics processing time |
| TOTAL_RUNSTATS | BIGINT | total_runstats - Total runtime statistics |
| TOTAL_REORG_TIME | BIGINT | total_reorg_time - Total reorganization time |
| TOTAL_REORG_PROC_TIME | BIGINT | total_reorg_proc_time - Total reorganization processing time |
| TOTAL_REORGS | BIGINT | total_reorgs - Total reorganizations |
| TOTAL_LOAD_TIME | BIGINT | total_load_time - Total load time |
| TOTAL_LOAD_PROC_TIME | BIGINT | total_load_proc_time - Total load processing time |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------------|-----------|---|
| TOTAL_LOADS | BIGINT | total_loads - Total loads |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups |
| THRESH_VIOLATIONS | BIGINT | thresh_violations - Number of threshold violations |
| NUM_LW_THRESH_EXCEEDED | BIGINT | num_lw_thresh_exceeded - Number of lock wait thresholds exceeded |
| LOCK_WAITS_GLOBAL | BIGINT | lock_waits_global - Lock waits global |
| LOCK_WAIT_TIME_GLOBAL | BIGINT | lock_wait_time_global - Lock wait time global |
| LOCK_TIMEOUTS_GLOBAL | BIGINT | lock_timeouts_global - Lock timeouts global |
| LOCK_ESCALS_MAXLOCKS | BIGINT | lock_escal_maxlocks - Number of maxlocks lock escalations |
| LOCK_ESCALS_LOCKLIST | BIGINT | lock_escal_locklist - Number of locklist lock escalations |
| LOCK_ESCALS_GLOBAL | BIGINT | lock_escal_global - Number of global lock escalations |
| RECLAIM_WAIT_TIME | BIGINT | reclaim_wait_time - Reclaim wait time |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME | BIGINT | spacemappage_reclaim_wait_time - Space map page reclaim wait time |
| CF_WAITS | BIGINT | cf_waits - Number of cluster caching facility waits |
| CF_WAIT_TIME | BIGINT | cf_wait_time - cluster caching facility wait time |
| POOL_DATA_GBP_L_READS | BIGINT | pool_data_gbp_l_reads - Group buffer pool data logical reads |
| POOL_DATA_GBP_P_READS | BIGINT | pool_data_gbp_p_reads - Group buffer pool data physical reads |
| POOL_DATA_LBP_PAGES_FOUND | BIGINT | pool_data_lbp_pages_found - Local buffer pool found data pages |
| POOL_DATA_GBP_INVALID_PAGES | BIGINT | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages |
| POOL_INDEX_GBP_L_READS | BIGINT | pool_index_gbp_l_reads - Group buffer pool index logical reads |
| POOL_INDEX_GBP_P_READS | BIGINT | pool_index_gbp_p_reads - Group buffer pool index physical reads |
| POOL_INDEX_LBP_PAGES_FOUND | BIGINT | pool_index_lbp_pages_found - Local buffer pool index pages found |
| POOL_INDEX_GBP_INVALID_PAGES | BIGINT | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages |
| POOL_XDA_GBP_L_READS | BIGINT | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------------------|-----------|---|
| POOL_XDA_GBP_P_READS | BIGINT | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |
| POOL_XDA_LBP_PAGES_FOUND | BIGINT | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |
| POOL_XDA_GBP_INVALID_PAGES | BIGINT | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| AUDIT_EVENTS_TOTAL | BIGINT | audit_events_total - Total audit events |
| AUDIT_FILE_WRITES_TOTAL | BIGINT | audit_file_writes_total - Total audit files written |
| AUDIT_FILE_WRITE_WAIT_TIME | BIGINT | audit_file_write_wait_time - Audit file write wait time |
| AUDIT_SUBSYSTEM_WAITS_TOTAL | BIGINT | audit_subsystem_waits_total - Total audit subsystem waits |
| AUDIT_SUBSYSTEM_WAIT_TIME | BIGINT | audit_subsystem_wait_time - Audit subsystem wait time |
| DIAGLOG_WRITES_TOTAL | BIGINT | diaglog_writes_total - Total diagnostic log file writes |
| DIAGLOG_WRITE_WAIT_TIME | BIGINT | diaglog_write_wait_time - Diagnostic log file write wait time |
| FCM_MESSAGE_RECVS_TOTAL | BIGINT | fcm_message_recvs_total - Total FCM message receives |
| FCM_MESSAGE_RECV_VOLUME | BIGINT | fcm_message_recv_volume - FCM message received volume |
| FCM_MESSAGE_RECV_WAIT_TIME | BIGINT | fcm_message_recv_wait_time - FCM message received wait time |
| FCM_MESSAGE_SENDS_TOTAL | BIGINT | fcm_message_sends_total - Total FCM message sends |
| FCM_MESSAGE_SEND_VOLUME | BIGINT | fcm_message_send_volume - FCM message send volume |
| FCM_MESSAGE_SEND_WAIT_TIME | BIGINT | fcm_message_send_wait_time - FCM message send wait time |
| FCM_TQ_RECVS_TOTAL | BIGINT | fcm_tq_recvs_total - FCM table queue receives total |
| FCM_TQ_RECV_VOLUME | BIGINT | fcm_tq_recv_volume - FCM table queue received volume |
| FCM_TQ_RECV_WAIT_TIME | BIGINT | fcm_tq_recv_wait_time - FCM table queue received wait time |
| FCM_TQ_SENDS_TOTAL | BIGINT | fcm_tq_sends_total - FCM table queue send total |
| FCM_TQ_SEND_VOLUME | BIGINT | fcm_tq_send_volume - FCM table queue send volume |
| FCM_TQ_SEND_WAIT_TIME | BIGINT | fcm_tq_send_wait_time - FCM table queue send wait time |
| TOTAL_ROUTINE_USER_CODE_PROC_TIME | BIGINT | total_routine_user_code_proc_time - Total routine user code processing time |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------------------|-----------|--|
| TOTAL_ROUTINE_USER_CODE_TIME | BIGINT | total_routine_user_code_time - Total routine user code time |
| TQ_TOT_SEND_SPILLS | BIGINT | tq_tot_send_spills - Total number of table queue buffers overflowed |
| EVMON_WAIT_TIME | BIGINT | evmon_wait_time - Event monitor wait time |
| EVMON_WAITS_TOTAL | BIGINT | evmon_waits_total - Event monitor total waits |
| TOTAL_EXTENDED_LATCH_WAIT_TIME | BIGINT | total_extended_latch_wait_time - Total extended latch wait time |
| TOTAL_EXTENDED_LATCH_WAITS | BIGINT | total_extended_latch_waits - Total extended latch waits |
| TOTAL_STATS_FABRICATION_TIME | BIGINT | total_stats_fabrication_time - Total statistics fabrication time |
| TOTAL_STATS_FABRICATION_PROC_TIME | BIGINT | total_stats_fabrication_proc_time - Total statistics fabrication processing time |
| TOTAL_STATS_FABRICATIONS | BIGINT | total_stats_fabrications - Total statistics fabrications |
| TOTAL_SYNC_RUNSTATS_TIME | BIGINT | total_sync_runstats_time - Total synchronous RUNSTATS time |
| TOTAL_SYNC_RUNSTATS_PROC_TIME | BIGINT | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time |
| TOTAL_SYNC_RUNSTATS | BIGINT | total_sync_runstats - Total synchronous RUNSTATS activities |
| TOTAL_DISP_RUN_QUEUE_TIME | BIGINT | total_disp_run_queue_time - Total dispatcher run queue time |
| TOTAL_PEDS | BIGINT | total_peds - Total partial early distincts |
| DISABLED_PEDS | BIGINT | disabled_peds - Disabled partial early distincts |
| POST_THRESHOLD_PEDS | BIGINT | post_threshold_peds - Partial early distincts threshold |
| TOTAL_PEAS | BIGINT | total_peas - Total partial early aggregations |
| POST_THRESHOLD_PEAS | BIGINT | post_threshold_peas - Partial early aggregation threshold |
| TQ_SORT_HEAP_REQUESTS | BIGINT | tq_sort_heap_requests - Table queue sort heap requests |
| TQ_SORT_HEAP_REJECTIONS | BIGINT | tq_sort_heap_rejections - Table queue sort heap rejections |
| POOL_QUEUED_ASYNC_DATA_REQS | BIGINT | pool_queued_async_data_reqs - Data prefetch requests |
| POOL_QUEUED_ASYNC_INDEX_REQS | BIGINT | pool_queued_async_index_reqs - Index prefetch requests |
| POOL_QUEUED_ASYNC_XDA_REQS | BIGINT | pool_queued_async_xda_reqs - XDA prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|---|-----------|--|
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_OTHER_REQS | BIGINT | pool_queued_async_other_reqs - Other requests handled by prefetchers |
| POOL_QUEUED_ASYNC_DATA_PAGES | BIGINT | pool_queued_async_data_pages - Data pages prefetch requests |
| POOL_QUEUED_ASYNC_INDEX_PAGES | BIGINT | pool_queued_async_index_pages - Index pages prefetch requests |
| POOL_QUEUED_ASYNC_XDA_PAGES | BIGINT | pool_queued_async_xda_pages - XDA pages prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES | BIGINT | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES | BIGINT | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_DATA_REQS | BIGINT | pool_failed_async_data_reqs - Failed data prefetch requests |
| POOL_FAILED_ASYNC_INDEX_REQS | BIGINT | pool_failed_async_index_reqs - Failed index prefetch requests |
| POOL_FAILED_ASYNC_XDA_REQS | BIGINT | pool_failed_async_xda_reqs - Failed XDA prefetch requests |
| POOL_FAILED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces |
| POOL_FAILED_ASYNC_OTHER_REQS | BIGINT | pool_failed_async_other_reqs - Failed non-prefetch requests |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| PREFETCH_WAITS | BIGINT | prefetch_waits - Prefetcher wait count |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------------------|-----------|---|
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool |
| COMM_EXIT_WAIT_TIME | BIGINT | comm_exit_wait_time - Communication buffer exit wait time |
| COMM_EXIT_WAITS | BIGINT | comm_exit_waits - Communication buffer exit number of waits |
| IDA_SEND_WAIT_TIME | BIGINT | ida_send_wait_time - Time spent waiting to send data |
| IDA_SENDS_TOTAL | BIGINT | ida_sends_total - Number of times data sent |
| IDA_SEND_VOLUME | BIGINT | ida_send_volume - Total data volume sent |
| IDA_RECV_WAIT_TIME | BIGINT | ida_recv_wait_time - Time spent waiting to receive data |
| IDA_RECVS_TOTAL | BIGINT | ida_recvs_total - Number of times data received |
| IDA_RECV_VOLUME | BIGINT | ida_recv_volume - Total data volume received |
| POOL_COL_L_READS | BIGINT | pool_col_l_reads - Buffer pool column-organized logical reads monitor element |
| POOL_TEMP_COL_L_READS | BIGINT | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads monitor element |
| POOL_COL_P_READS | BIGINT | pool_col_p_reads - Buffer pool column-organized physical reads monitor element |
| POOL_TEMP_COL_P_READS | BIGINT | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads monitor element |
| POOL_COL_LBP_PAGES_FOUND | BIGINT | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found monitor element |
| POOL_COL_WRITES | BIGINT | pool_col_writes - Buffer pool column-organized writes monitor element |
| POOL_COL_GBP_L_READS | BIGINT | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads monitor element |
| POOL_COL_GBP_P_READS | BIGINT | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads monitor element |
| POOL_COL_GBP_INVALID_PAGES | BIGINT | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages monitor element |
| POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool monitor element |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------------|-----------|--|
| POOL_QUEUED_ASYNC_COL_REQS | BIGINT | pool_queued_async_col_reqs - Column-organized prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_COL_REQS | BIGINT | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests monitor element |
| POOL_QUEUED_ASYNC_COL_PAGES | BIGINT | pool_queued_async_col_pages - Column-organized page prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_COL_PAGES | BIGINT | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests monitor element |
| POOL_FAILED_ASYNC_COL_REQS | BIGINT | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests monitor element |
| POOL_FAILED_ASYNC_TEMP_COL_REQS | BIGINT | pool_failed_async_col_reqs - Failed column-organized prefetch requests monitor element |
| TOTAL_COL_TIME | BIGINT | total_col_time - Total column-organized time monitor element |
| TOTAL_COL_PROC_TIME | BIGINT | total_col_proc_time - Total column-organized processing time monitor element |
| TOTAL_COL_EXECUTIONS | BIGINT | total_col_executions - Total column-organized executions monitor element |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted monitor element |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted monitor element |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated monitor element |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total Hash Joins monitor element |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total Hash Loops monitor element |
| HASH_JOIN_OVERFLOW | BIGINT | hash_join_overflows - Hash Join Overflows monitor element |
| HASH_JOIN_SMALL_OVERFLOW | BIGINT | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| POST_SHRTHRESHOLD_HASH_JOINS | BIGINT | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP Functions monitor element |
| OLAP_FUNC_OVERFLOW | BIGINT | olap_func_overflows - OLAP Function Overflows monitor element |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------------------|-----------|---|
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL Statements Attempted monitor element |
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed Statement Operations monitor element |
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL Statements Executed monitor element |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| MERGE_SQL_STMTS | BIGINT | merge_sql_stmts - Merge SQL statements executed monitor element |
| XQUERY_STMTS | BIGINT | xquery_stmts - XQuery Statements Attempted monitor element |
| IMPLICIT_REBINDS | BIGINT | implicit_rebinds - number of implicit rebinds monitor element |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/Precompiles Attempted monitor element |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal Rows Inserted monitor element |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal Rows Updated monitor element |
| CALL_SQL_STMTS | BIGINT | call_sql_stmts - CALL SQL statements executed monitor element |
| POST_THRESHOLD_HASH_JOINS | BIGINT | post_threshold_hash_joins - Hash Join Threshold monitor element |
| POOL_DATA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|---|-----------|--|
| POOL_COL_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_READ_TIME | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_WRITE_TIME | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| TOTAL_HASH_GRPBY | BIGINT | total_hash_grpbys - Total hash group by operations |
| HASH_GRPBY_OVERFLOW | BIGINT | hash_grpby_overflows - Hash group by overflows |
| POST_THRESHOLD_HASH_GRPBY | BIGINT | post_threshold_hash_grpbys - Hash group by threshold |
| POST_THRESHOLD_OLAP_FUNCS | BIGINT | post_threshold_olap_funcs - OLAP function threshold monitor element |
| POST_THRESHOLD_COL_VECTOR_CONSUMERS | BIGINT | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| TOTAL_COL_VECTOR_CONSUMERS | BIGINT | total_col_vector_consumers - Total columnar_vector consumers |
| TOTAL_BACKUP_TIME | BIGINT | total_backup_time - Total elapsed time for online backups |
| TOTAL_BACKUP_PROC_TIME | BIGINT | total_backup_proc_time - Total non-wait time for online backups |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------------|-----------|--|
| TOTAL_BACKUPS | BIGINT | total_backups - Total backups |
| TOTAL_INDEX_BUILD_TIME | BIGINT | total_index_build_time - Total time spent building indexes due to index creation |
| TOTAL_INDEX_BUILD_PROC_TIME | BIGINT | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation |
| TOTAL_INDEXES_BUILT | BIGINT | total_indexes_built - Total number of indexes built |
| FCM_TQ_RECV_WAITS_TOTAL | BIGINT | fcm_tq_recv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| FCM_MESSAGE_RECV_WAITS_TOTAL | BIGINT | fcm_message_recv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| FCM_TQ_SEND_WAITS_TOTAL | BIGINT | fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |
| FCM_MESSAGE_SEND_WAITS_TOTAL | BIGINT | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| FCM_SEND_WAITS_TOTAL | BIGINT | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| FCM_RECV_WAITS_TOTAL | BIGINT | fcm_recv_waits_total - Number of times spent waiting to receive data through FCM monitor element |
| EXT_TABLE_RECV_WAIT_TIME | BIGINT | ext_table_recv_wait_time - Total agent wait time for external table readers monitor element |
| EXT_TABLE_RECVS_TOTAL | BIGINT | ext_table_recvs_total - Total row batches received from external table readers monitor element |
| EXT_TABLE_RECV_VOLUME | BIGINT | ext_table_recv_volume - Total data received from external table readers monitor element |
| EXT_TABLE_READ_VOLUME | BIGINT | ext_table_read_volume - Total data read by external table readers monitor element |
| EXT_TABLE_SEND_WAIT_TIME | BIGINT | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| EXT_TABLE_SENDS_TOTAL | BIGINT | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| EXT_TABLE_SEND_VOLUME | BIGINT | ext_table_send_volume - Total data sent to external table writers monitor element |
| EXT_TABLE_WRITE_VOLUME | BIGINT | ext_table_write_volume - Total data written by external table writers monitor element |
| COL_VECTOR_CONSUMER_OVERFLOWES | BIGINT | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |

Table 147. Information returned for MON_GET_ROUTINE (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------------|-----------|--|
| TOTAL_COL_SYNOPSIS_TIME | BIGINT | total_col_synopsis_time - Total column-organized synopsis table time monitor element |
| TOTAL_COL_SYNOPSIS_PROC_TIME | BIGINT | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| TOTAL_COL_SYNOPSIS_EXECUTIONS | BIGINT | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |
| COL_SYNOPSIS_ROWS_INSERTED | BIGINT | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| LOB_PREFETCH_WAIT_TIME | BIGINT | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| LOB_PREFETCH_REQS | BIGINT | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| FED_ROWS_DELETED | BIGINT | fed_rows_deleted - Rows deleted by a federation system monitor element |
| FED_ROWS_INSERTED | BIGINT | fed_rows_inserted - Rows inserted by a federation system monitor element |
| FED_ROWS_UPDATED | BIGINT | fed_rows_updated - Rows updated by a federation system monitor element |
| FED_ROWS_READ | BIGINT | fed_rows_read - Rows read by a federation system monitor element |
| FED_WAIT_TIME | BIGINT | fed_wait_time - Time spent by a federation server monitor element |
| FED_WAITS_TOTAL | BIGINT | fed_waits_total - Total number of execution times for a federation server monitor element |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Insert count for SQL workspace monitor element |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section lookups - SQL workspace access count monitor element |
| ADM_OVERFLOW | BIGINT | Reserved for future use. |
| ADM_BYPASS_ACT_TOTAL | BIGINT | Reserved for future use. |

Usage notes

Use the MON_GET_ROUTINE table function to identify the most expensive routines on the database server.

This table function returns one row of metrics for each routine or trigger and each member that matches the input arguments. Input argument values are complementary. Metrics returned are aggregates of all executions of the routine on that member. No aggregation across members is performed. However, an aggregation across members is possible through SQL queries (as shown in the Examples section). If a routine executes subroutines, the work done in the subroutines is included in the metrics of the parent routine.

Routine monitoring data collection must be explicitly enabled using the `mon_rtn_data` database configuration parameter. If this configuration parameter is set to `NONE`, no information is returned.

The counters and time-spent monitor elements returned by this table function are controlled with the `COLLECT REQUEST METRICS` clause on service classes and the `mon_req_metrics` database configuration parameter at the database level. If neither control is enabled, the counters and time-spent monitor elements reported are 0.

When the package for a dynamically prepared compound SQL statements is removed from the package cache, information for this routine is no longer reported by `MON_GET_ROUTINE` function. Similarly, when a routine or trigger is dropped, information about the routine or trigger is no longer reported.

Any routines that were not executed during the previous 24 hour period are pruned from memory and not returned.

Examples

1. List the highest CPU consuming routines of any type in module MOD1.

```
SELECT ROUTINE_TYPE, ROUTINE_SCHEMA, ROUTINE_NAME, SPECIFIC_NAME, TOTAL_CPU_TIME
FROM TABLE(MON_GET_ROUTINE(NULL, NULL, 'MOD1', NULL, NULL))
AS T ORDER BY TOTAL_CPU_TIME DESC
```

returns

| ROUTINE_TYPE | ROUTINE_SCHEMA | ROUTINE_NAME | SPECIFIC_NAME | TOTAL_CPU_TIME |
|--------------|----------------|----------------------------|--------------------|----------------|
| F | DRICARD | FUNCTION3 | FUNCTION3 | 19425 |
| F | DRICARD | FUNCTION4 | FUNCTION4 | 5780 |
| P | DRICARD | P1 | SQL120801142627900 | 4685 |
| C | SYSIBMINTERNAL | COMPILED_ANON_BLOCK_INVOKE | SQL120801153841490 | 3471 |
| P | SYSPROC | SYSINSTALLOBJECTS | SYSINSTALLOBJECTS | 1158 |
| F | DRICARD | FUNCTION1 | FUNCTION1 | 2632 |
| F | DRICARD | FUNCTION2 | FUNCTION2 | 2029 |

7 record(s) selected.

2. List aggregate monitor data for all stored procedures.

```
SELECT * FROM TABLE(MON_GET_ROUTINE('P', NULL, NULL, NULL, -2)) AS T
```
3. List aggregate monitor data for routines of all types.

```
SELECT * FROM TABLE(MON_GET_ROUTINE(NULL, NULL, NULL, NULL, -2)) AS T
```
4. List aggregate monitor data for all procedures named PROC1 in schema TEST.

```
SELECT * FROM TABLE(MON_GET_ROUTINE('P', 'TEST', NULL, 'PROC1', -2)) AS T
```
5. List aggregate monitor data for overloaded procedure PROC1 with specific name PROC1_OVERLOAD in schema TEST.

```
SELECT * FROM TABLE(MON_GET_ROUTINE('SP', 'TEST', NULL, 'PROC1_OVERLOAD', -2))
AS T
```
6. List aggregate monitor data for the anonymous block with executable id x'010000000000000052010000000000001000000010020120822205618607103'.

```
SELECT * FROM TABLE(MON_GET_ROUTINE
('A', 'MYSCHEMA', NULL, 'SQL181500027522310', -1)) AS T
WHERE
DYN_COMPOUND_EXEC_ID = x'010000000000000052010000000000001000000010020120822205618607103'
```

MON_GET_ROUTINE_DETAILS table function - get aggregated routine execution metrics as an XML document

The `MON_GET_ROUTINE_DETAILS` table function returns aggregated execution metrics for procedures, external functions, compiled functions, compiled triggers, and anonymous blocks invoked since the database was activated.

The metrics are returned in an XML document.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- SQLADM authority
- DBADM authority

Default PUBLIC privilege

None

Syntax

```
►►MON_GET_ROUTINE_DETAILS—(—routine_type—,—routine_schema—,——————►  
►—routine_module_name—,—routine_name—,—member—)——————►◄
```

The schema is SYSPROC.

Routine parameters

routine_type

An input parameter of type CHAR(2) that specifies the type of routine or compiled trigger for which data is returned:

- "P" for a procedure
- "SP" for the specific name of a procedure
- "F" for a compiled function
- "SF" for the specific name of a compiled function
- "T" for a compiled trigger
- "C" for an anonymous block

If the "P", "F", "T", or "C" routine type is specified and no input name is specified, all routines of the given type will be returned. If the routine type is "SP" or "SF", the specific name of the routine must be provided.

All routines of all types will be returned if the routine type is NULL, an empty string, or blanks.

routine_schema

An input parameter of type VARCHAR(128) that specifies the schema for the routine or trigger. For dynamically prepared compound SQL statements and PL/SQL anonymous blocks, the schema can be determined using the MON_GET_SECTION_ROUTINES table function. Use NULL or an empty string to return the routines and triggers in all schemas. This parameter is case sensitive.

routine_module_name

An input parameter of type VARCHAR(128) that specifies the name of the module for the input routine, if applicable. Use NULL or an empty string to return the routines in all modules. This parameter is case sensitive.

routine_name

An input parameter of type VARCHAR(128) that specifies the name of the

routine. If the input parameter is "SP" or "SF", the specific name of the routine should be provided. For dynamically prepared compound SQL statements and PL/SQL anonymous blocks, the name can be determined using the MON_GET_SECTION_ROUTINES table function. Use NULL or an empty string to return all routines matching the other input parameters. This parameter is case sensitive.

member

An input parameter of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is implicitly set.

Information returned

Table 148. Information returned for MON_GET_ROUTINE_DETAILS

| Column name | Data type | Description or corresponding monitor element |
|----------------------|--------------------------|---|
| ROUTINE_TYPE | CHAR(1) | routine_type - Routine type monitor element |
| ROUTINE_SCHEMA | VARCHAR(128) | routine_schema - Routine schema monitor element |
| ROUTINE_MODULE_NAME | VARCHAR(128) | routine_module_name - Routine module name monitor element |
| ROUTINE_NAME | VARCHAR(128) | routine_name - Routine name monitor element |
| SPECIFIC_NAME | VARCHAR(128) | specific_name - Specific name monitor element |
| DYN_COMPOUND_EXEC_ID | VARCHAR(32) FOR BIT DATA | dyn_compound_exec_id - Dynamic compound statement executable identifier monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| ROUTINE_ID | INTEGER | routine_id - Routine ID monitor element |
| SUBROUTINE_ID | INTEGER | subroutine_id - Subroutine identifier monitor element |
| LIB_ID | BIGINT | lib_id - Library identifier monitor element |
| DETAILS | BLOB(1M) | XML document that contains the monitor information for the routine. See Table 2 for a description of the elements in this document. |

The following example shows the structure of the XML document that is returned in the DETAILS column.

```
<db2_routine xmlns="http://www.ibm.com/xmlns/prod/db2/mon" release="100102000">
  <routine_type>P</routine_type>
  <routine_schema>TEST</routine_schema>
  <routine_module></routine_module>
  <routine_name>PROC1</routine_name>
  ...
</db2_routine>
```

```

<system_metrics release="100102000">
  <act_aborted_total>5</act_aborted_total>
</system_metrics>
</db2_routine>

```

See `sqllib/misc/DB2MonRoutines.xsd` for the full schema.

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS

| Element name | Data type | Description or corresponding monitor element |
|-------------------------------|-----------------------|--|
| act_aborted_total | xs:long | act_aborted_total - Total aborted activities |
| act_completed_total | xs:long | act_completed_total - Total completed activities |
| act_rejected_total | xs:long | act_rejected_total - Total rejected activities |
| act_rqsts_total | xs:long | act_rqsts_total - Total activity requests |
| adm_bypass_act_total | xs:long | Reserved for future use. |
| adm_overflows | xs:long | Reserved for future use. |
| app_rqsts_completed_total | xs:long | app_rqsts_completed_total - Total application requests completed |
| appl_section_inserts | xs:nonNegativeInteger | appl_section_inserts - Insert count for SQL workspace monitor element |
| appl_section_lookups | xs:nonNegativeInteger | appl_section_lookups - SQL workspace access count monitor element |
| audit_events_total | xs:long | audit_events_total - Total audit events |
| audit_file_write_wait_time | xs:long | audit_file_write_wait_time - Audit file write wait time |
| audit_file_writes_total | xs:long | audit_file_writes_total - Total audit files written |
| audit_subsystem_wait_time | xs:long | audit_subsystem_wait_time - Audit subsystem wait time |
| audit_subsystem_waits_total | xs:long | audit_subsystem_waits_total - Total audit subsystem waits |
| binds_precompiles | xs:long | binds_precompiles - Binds/Precompiles Attempted monitor element |
| call_sql_stmts | xs:long | call_sql_stmts - CALL SQL statements executed monitor element |
| cat_cache_inserts | xs:long | cat_cache_inserts - Catalog cache inserts |
| cat_cache_lookups | xs:long | cat_cache_lookups - Catalog cache lookups |
| cf_wait_time | xs:long | cf_wait_time - cluster caching facility wait time |
| cf_waits | xs:long | cf_waits - Number of cluster caching facility waits |
| col_synopsis_rows_inserted | xs:nonNegativeInteger | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| col_vector_consumer_overflows | xs:long | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |
| comm_exit_wait_time | xs:long | comm_exit_wait_time - Communication buffer exit wait time |

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|--------------------------|-----------------------|---|
| comm_exit_waits | xs:long | comm_exit_waits - Communication buffer exit number of waits |
| ddl_sql_stmts | xs:long | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| deadlocks | xs:long | deadlocks - Deadlocks detected |
| diaglog_write_wait_time | xs:long | diaglog_write_wait_time - Diagnostic log file write wait time |
| diaglog_writes_total | xs:long | diaglog_writes_total - Total diagnostic log file writes |
| direct_read_reqs | xs:long | direct_read_reqs - Direct read requests |
| direct_read_time | xs:long | direct_read_time - Direct read time |
| direct_reads | xs:long | direct_reads - Direct reads from database |
| direct_write_reqs | xs:long | direct_write_reqs - Direct write requests |
| direct_write_time | xs:long | direct_write_time - Direct write time |
| direct_writes | xs:long | direct_writes - Direct writes to database |
| disabled_peds | xs:long | disabled_peds - Disabled partial early distincts |
| dyn_compound_exec_id | xs:hexBinary(32) | dyn_compound_exec_id - Dynamic compound statement executable identifier monitor element |
| dynamic_sql_stmts | xs:long | dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element |
| evmon_wait_time | xs:long | evmon_wait_time - Event monitor wait time |
| evmon_waits_total | xs:long | evmon_waits_total - Event monitor total waits |
| exec_list_cleanup_time | xs:dateTime | exec_list_cleanup_time - Execution list cleanup time monitor element |
| exec_list_mem_exceeded | xs:string(1) | exec_list_mem_exceeded - Execution list memory exceeded monitor element |
| ext_table_read_volume | xs:nonNegativeInteger | ext_table_read_volume - Total data read by external table readers monitor element |
| ext_table_rcv_volume | xs:nonNegativeInteger | ext_table_rcv_volume - Total data received from external table readers monitor element |
| ext_table_rcv_wait_time | xs:nonNegativeInteger | ext_table_rcv_wait_time - Total agent wait time for external table readers monitor element |
| ext_table_rcvs_total | xs:nonNegativeInteger | ext_table_rcvs_total - Total row batches received from external table readers monitor element |
| ext_table_send_volume | xs:nonNegativeInteger | ext_table_send_volume - Total data sent to external table writers monitor element |
| ext_table_send_wait_time | xs:nonNegativeInteger | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| ext_table_sends_total | xs:nonNegativeInteger | ext_table_sends_total - Total row batches sent to external table writers monitor element |

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|------------------------------|-----------------------|--|
| ext_table_write_volume | xs:nonNegativeInteger | ext_table_write_volume - Total data written by external table writers monitor element |
| failed_sql_stmts | xs:long | failed_sql_stmts - Failed Statement Operations monitor element |
| fcm_message_rcv_volume | xs:long | fcm_message_rcv_volume - FCM message received volume |
| fcm_message_rcv_wait_time | xs:long | fcm_message_rcv_wait_time - FCM message received wait time |
| fcm_message_rcv_waits_total | xs:nonNegativeInteger | fcm_message_rcv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| fcm_message_rcvs_total | xs:long | fcm_message_rcvs_total - Total FCM message receives |
| fcm_message_send_volume | xs:long | fcm_message_send_volume - FCM message send volume |
| fcm_message_send_wait_time | xs:long | fcm_message_send_wait_time - FCM message send wait time |
| fcm_message_send_waits_total | xs:nonNegativeInteger | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| fcm_message_sends_total | xs:long | fcm_message_sends_total - Total FCM message sends |
| fcm_rcv_volume | xs:long | fcm_rcv_volume - FCM received volume |
| fcm_rcv_wait_time | xs:long | fcm_rcv_wait_time - FCM received wait time |
| fcm_rcv_waits_total | xs:nonNegativeInteger | fcm_rcv_waits_total - Number of times spent waiting to receive data through FCM monitor element |
| fcm_rcvs_total | xs:long | fcm_rcvs_total - FCM receives total |
| fcm_send_volume | xs:long | fcm_send_volume - FCM send volume |
| fcm_send_wait_time | xs:long | fcm_send_wait_time - FCM send wait time |
| fcm_sends_total | xs:long | fcm_sends_total - FCM sends total |
| fcm_tq_rcv_volume | xs:long | fcm_tq_rcv_volume - FCM table queue received volume |
| fcm_tq_rcv_wait_time | xs:long | fcm_tq_rcv_wait_time - FCM table queue received wait time |
| fcm_tq_rcv_waits_total | xs:nonNegativeInteger | fcm_tq_rcv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| fcm_tq_rcvs_total | xs:long | fcm_tq_rcvs_total - FCM table queue receives total |
| fcm_tq_send_volume | xs:long | fcm_tq_send_volume - FCM table queue send volume |
| fcm_tq_send_wait_time | xs:long | fcm_tq_send_wait_time - FCM table queue send wait time |

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|---------------------------|-----------------------|---|
| fcm_tq_sends_total | xs:long | fcm_tq_sends_total - FCM table queue send total |
| fed_rows_deleted | xs:nonNegativeInteger | fed_rows_deleted - Rows deleted by a federation system monitor element |
| fed_rows_inserted | xs:nonNegativeInteger | fed_rows_inserted - Rows inserted by a federation system monitor element |
| fed_rows_read | xs:nonNegativeInteger | fed_rows_read - Rows read by a federation system monitor element |
| fed_rows_updated | xs:nonNegativeInteger | fed_rows_updated - Rows updated by a federation system monitor element |
| fed_wait_time | xs:nonNegativeInteger | fed_wait_time - Time spent by a federation server monitor element |
| fed_waits_total | xs:nonNegativeInteger | fed_waits_total - Total number of execution times for a federation server monitor element |
| hash_grpby_overflows | xs:long | hash_grpby_overflows - Hash group by overflows |
| hash_join_overflows | xs:long | hash_join_overflows - Hash Join Overflows monitor element |
| hash_join_small_overflows | xs:long | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| ida_rcv_volume | xs:long | ida_rcv_volume - Total data volume received |
| ida_rcv_wait_time | xs:long | ida_rcv_wait_time - Time spent waiting to receive data |
| ida_rcvs_total | xs:long | ida_rcvs_total - Number of times data received |
| ida_send_volume | xs:long | ida_send_volume - Total data volume sent |
| ida_send_wait_time | xs:long | ida_send_wait_time - Time spent waiting to send data |
| ida_sends_total | xs:long | ida_sends_total - Number of times data sent |
| implicit_rebinds | xs:long | implicit_rebinds - number of implicit rebinds monitor element |
| int_commits | xs:long | int_commits - Internal commits |
| int_rollbacks | xs:long | int_rollbacks - Internal rollbacks |
| int_rows_deleted | xs:long | int_rows_deleted - Internal Rows Deleted monitor element |
| int_rows_inserted | xs:long | int_rows_inserted - Internal Rows Inserted monitor element |
| int_rows_updated | xs:long | int_rows_updated - Internal Rows Updated monitor element |
| lib_id | xs:long | lib_id - Library identifier monitor element |
| lob_prefetch_reqs | xs:nonNegativeInteger | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| lob_prefetch_wait_time | xs:nonNegativeInteger | lob_prefetch_wait_time - LOB prefetch wait time monitor element |

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|---------------------------------------|-----------------------|---|
| lock_escals | xs:long | lock_escals - Number of lock escalations |
| lock_escals_global | xs:long | lock_escals_global - Number of global lock escalations |
| lock_escals_locklist | xs:long | lock_escals_locklist - Number of locklist lock escalations |
| lock_escals_maxlocks | xs:long | lock_escals_maxlocks - Number of maxlocks lock escalations |
| lock_timeouts | xs:long | lock_timeouts - Number of lock timeouts |
| lock_timeouts_global | xs:long | lock_timeouts_global - Lock timeouts global |
| lock_wait_time | xs:long | lock_wait_time - Time waited on locks |
| lock_wait_time_global | xs:long | lock_wait_time_global - Lock wait time global |
| lock_waits | xs:long | lock_waits - Lock waits |
| lock_waits_global | xs:long | lock_waits_global - Lock waits global |
| log_buffer_wait_time | xs:long | log_buffer_wait_time - Log buffer wait time |
| log_disk_wait_time | xs:long | log_disk_wait_time - Log disk wait time |
| log_disk_waits_total | xs:long | log_disk_waits_total - Total log disk waits |
| member | xs:short | member - Database member monitor element |
| merge_sql_stmts | xs:long | merge_sql_stmts - Merge SQL statements executed monitor element |
| num_log_buffer_full | xs:long | num_log_buffer_full - Number of times full log buffer caused agents to wait |
| num_lw_thresh_exceeded | xs:long | num_lw_thresh_exceeded - Number of lock wait thresholds exceeded |
| olap_func_overflows | xs:long | olap_func_overflows - OLAP Function Overflows monitor element |
| pkg_cache_inserts | xs:long | pkg_cache_inserts - Package cache inserts |
| pkg_cache_lookups | xs:long | pkg_cache_lookups - Package cache lookups |
| pool_col_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool monitor element |
| pool_col_gbp_invalid_pages | xs:nonNegativeInteger | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages monitor element |
| pool_col_gbp_l_reads | xs:nonNegativeInteger | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads monitor element |
| pool_col_gbp_p_reads | xs:nonNegativeInteger | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads monitor element |
| pool_col_l_reads | xs:nonNegativeInteger | pool_col_l_reads - Buffer pool column-organized logical reads monitor element |

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|--|-----------------------|--|
| pool_col_lbp_pages_found | xs:nonNegativeInteger | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found monitor element |
| pool_col_p_reads | xs:nonNegativeInteger | pool_col_p_reads - Buffer pool column-organized physical reads monitor element |
| pool_col_writes | xs:nonNegativeInteger | pool_col_writes - Buffer pool column-organized writes monitor element |
| pool_data_gbp_indep_pages_found_in_lbp | xs:long | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool |
| pool_data_gbp_invalid_pages | xs:long | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages |
| pool_data_gbp_l_reads | xs:long | pool_data_gbp_l_reads - Group buffer pool data logical reads |
| pool_data_gbp_p_reads | xs:long | pool_data_gbp_p_reads - Group buffer pool data physical reads |
| pool_data_l_reads | xs:long | pool_data_l_reads - Buffer pool data logical reads |
| pool_data_lbp_pages_found | xs:long | pool_data_lbp_pages_found - Local buffer pool found data pages |
| pool_data_p_reads | xs:long | pool_data_p_reads - Buffer pool data physical reads |
| pool_data_writes | xs:long | pool_data_writes - Buffer pool data writes |
| pool_failed_async_col_reqs | xs:nonNegativeInteger | pool_failed_async_col_reqs - Failed column-organized prefetch requests monitor element |
| pool_failed_async_data_reqs | xs:long | pool_failed_async_data_reqs - Failed data prefetch requests |
| pool_failed_async_index_reqs | xs:long | pool_failed_async_index_reqs - Failed index prefetch requests |
| pool_failed_async_other_reqs | xs:long | pool_failed_async_other_reqs - Failed non-prefetch requests |
| pool_failed_async_temp_data_reqs | xs:long | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces |
| pool_failed_async_temp_index_reqs | xs:long | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces |
| pool_failed_async_temp_xda_reqs | xs:long | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces |
| pool_failed_async_xda_reqs | xs:long | pool_failed_async_xda_reqs - Failed XDA prefetch requests |
| pool_failed_async_temp_col_reqs | xs:nonNegativeInteger | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests monitor element |

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|---|-----------------------|--|
| pool_index_gbp_indep_pages_found_in_lbp | xs:long | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool |
| pool_index_gbp_invalid_pages | xs:long | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages |
| pool_index_gbp_l_reads | xs:long | pool_index_gbp_l_reads - Group buffer pool index logical reads |
| pool_index_gbp_p_reads | xs:long | pool_index_gbp_p_reads - Group buffer pool index physical reads |
| pool_index_l_reads | xs:long | pool_index_l_reads - Buffer pool index logical reads |
| pool_index_lbp_pages_found | xs:long | pool_index_lbp_pages_found - Local buffer pool index pages found |
| pool_index_p_reads | xs:long | pool_index_p_reads - Buffer pool index physical reads |
| pool_index_writes | xs:long | pool_index_writes - Buffer pool index writes |
| pool_queued_async_col_pages | xs:nonNegativeInteger | pool_queued_async_col_pages - Column-organized page prefetch requests monitor element |
| pool_queued_async_col_reqs | xs:nonNegativeInteger | pool_queued_async_col_reqs - Column-organized prefetch requests monitor element |
| pool_queued_async_data_pages | xs:long | pool_queued_async_data_pages - Data pages prefetch requests |
| pool_queued_async_data_reqs | xs:long | pool_queued_async_data_reqs - Data prefetch requests |
| pool_queued_async_index_pages | xs:long | pool_queued_async_index_pages - Index pages prefetch requests |
| pool_queued_async_index_reqs | xs:long | pool_queued_async_index_reqs - Index prefetch requests |
| pool_queued_async_other_reqs | xs:long | pool_queued_async_other_reqs - Other requests handled by prefetchers |
| pool_queued_async_xda_pages | xs:long | pool_queued_async_xda_pages - XDA pages prefetch requests |
| pool_queued_async_xda_reqs | xs:long | pool_queued_async_xda_reqs - XDA prefetch requests |
| pool_queued_async_temp_data_pages | xs:long | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces |
| pool_queued_async_temp_data_reqs | xs:long | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces |
| pool_queued_async_temp_index_pages | xs:long | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces |
| pool_queued_async_temp_index_reqs | xs:long | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces |

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|---------------------------------------|-----------------------|--|
| pool_queued_async_temp_xda_pages | xs:long | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces |
| pool_queued_async_temp_xda_reqs | xs:long | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces |
| pool_queued_async_temp_col_reqs | xs:nonNegativeInteger | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests monitor element |
| pool_queued_async_temp_col_pages | xs:nonNegativeInteger | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests monitor element |
| pool_read_time | xs:long | pool_read_time - Total buffer pool physical read time |
| pool_temp_col_l_reads | xs:nonNegativeInteger | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads monitor element |
| pool_temp_col_p_reads | xs:nonNegativeInteger | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads monitor element |
| pool_temp_data_l_reads | xs:long | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| pool_temp_data_p_reads | xs:long | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| pool_temp_index_l_reads | xs:long | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| pool_temp_index_p_reads | xs:long | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| pool_temp_xda_l_reads | xs:long | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |
| pool_temp_xda_p_reads | xs:long | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| pool_write_time | xs:long | pool_write_time - Total buffer pool physical write time |
| pool_xda_gbp_indep_pages_found_in_lbp | xs:long | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool |
| pool_xda_gbp_invalid_pages | xs:long | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| pool_xda_gbp_l_reads | xs:long | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |
| pool_xda_gbp_p_reads | xs:long | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |
| pool_xda_l_reads | xs:long | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| pool_xda_lbp_pages_found | xs:long | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|-------------------------------------|-----------------------|--|
| pool_xda_p_reads | xs:long | pool_xda_p_reads - Buffer pool XDA data physical reads |
| pool_xda_writes | xs:long | pool_xda_writes - Buffer pool XDA data writes |
| post_shrthreshold_hash_joins | xs:long | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| post_shrthreshold_sorts | xs:long | post_shrthreshold_sorts - Post shared threshold sorts |
| post_threshold_col_vector_consumers | xs:long | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| post_threshold_hash_grpbys | xs:long | post_threshold_hash_grpbys - Hash group by threshold
post_threshold_hash_grpbys - Hash group by threshold |
| post_threshold_hash_joins | xs:long | post_threshold_hash_joins - Hash Join Threshold monitor element |
| post_threshold_olap_funcs | xs:long | post_threshold_olap_funcs - OLAP function threshold monitor element |
| post_threshold_peas | xs:long | post_threshold_peas - Partial early aggregation threshold |
| post_threshold_peds | xs:long | post_threshold_peds - Partial early distincts threshold |
| post_threshold_sorts | xs:long | post_threshold_sorts - Post threshold sorts |
| prefetch_wait_time | xs:long | prefetch_wait_time - Time waited for prefetch |
| prefetch_waits | xs:long | prefetch_waits - Prefetcher wait count |
| fcm_tq_send_waits_total | xs:nonNegativeInteger | fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |
| reclaim_wait_time | xs:long | reclaim_wait_time - Reclaim wait time |
| routine_id | xs:int | routine_id - Routine ID monitor element |
| routine_module_name | xs:string(128) | routine_module_name - Routine module name monitor element |
| routine_name | xs:string(128) | routine_name - Routine name monitor element |
| routine_schema | xs:string(128) | routine_schema - Routine schema monitor element |
| routine_type | xs:string(1) | routine_type - Routine type monitor element |
| rows_deleted | xs:long | rows_deleted - Rows deleted monitor element |
| rows_inserted | xs:long | rows_inserted - Rows inserted monitor element |
| rows_modified | xs:long | rows_modified - Rows modified |
| rows_read | xs:long | rows_read - Rows read |
| rows_returned | xs:long | rows_returned - Rows returned |

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|--------------------------------|-----------------------|--|
| rows_updated | xs:long | rows_updated - Rows updated monitor element |
| rqsts_completed_total | xs:long | rqsts_completed_total - Total requests completed |
| select_sql_stmts | xs:long | select_sql_stmts - Select SQL Statements Executed monitor element |
| sort_overflows | xs:long | sort_overflows - Sort overflows |
| spacemappage_reclaim_wait_time | xs:long | spacemappage_reclaim_wait_time - Space map page reclaim wait time |
| specific_name | xs:string(128) | specific_name - Specific name monitor element |
| static_sql_stmts | xs:long | static_sql_stmts - Static SQL Statements Attempted monitor element |
| subroutine_id | xs:int | subroutine_id - Subroutine identifier monitor element |
| thresh_violations | xs:long | thresh_violations - Number of threshold violations |
| total_act_time | xs:long | total_act_time - Total activity time |
| total_act_wait_time | xs:long | total_act_wait_time - Total activity wait time |
| total_app_commits | xs:long | total_app_commits - Total application commits |
| total_app_rollback | xs:long | total_app_rollback - Total application rollbacks |
| total_app_rqst_time | xs:long | total_app_rqst_time - Total application request time |
| total_app_section_executions | xs:long | total_app_section_executions - Total application section executions |
| total_backup_proc_time | xs:long | total_backup_proc_time - Total non-wait time for online backups |
| total_backup_time | xs:long | total_backup_time - Total elapsed time for online backups |
| total_backups | xs:long | total_backups - Total backups |
| total_col_executions | xs:nonNegativeInteger | total_col_executions - Total column-organized executions monitor element |
| total_col_proc_time | xs:nonNegativeInteger | total_col_proc_time - Total column-organized processing time monitor element |
| total_col_synopsis_executions | xs:nonNegativeInteger | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |
| total_col_synopsis_proc_time | xs:nonNegativeInteger | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| total_col_synopsis_time | xs:nonNegativeInteger | total_col_synopsis_time - Total column-organized synopsis table time monitor element |

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|----------------------------------|-----------------------|--|
| total_col_time | xs:nonNegativeInteger | total_col_time - Total column-organized time monitor element |
| total_col_vector_consumers | xs:long | total_col_vector_consumers - Total columnar_vector consumers |
| total_commit_proc_time | xs:long | total_commit_proc_time - Total commits processing time |
| total_commit_time | xs:long | total_commit_time - Total commit time |
| total_compilations | xs:long | total_compilations - Total compilations |
| total_compile_proc_time | xs:long | total_compile_proc_time - Total compile processing time |
| total_compile_time | xs:long | total_compile_time - Total compile time |
| total_cpu_time | xs:long | total_cpu_time - Total CPU time. |
| total_disp_run_queue_time | xs:long | total_disp_run_queue_time - Total dispatcher run queue time |
| total_extended_latch_waits | xs:long | total_extended_latch_waits - Total extended latch waits |
| total_extended_latch_wait_time | xs:long | total_extended_latch_wait_time - Total extended latch wait time |
| total_hash_grpbys | xs:long | total_hash_grpbys - Total hash group by operations |
| total_hash_joins | xs:long | total_hash_joins - Total Hash Joins monitor element |
| total_hash_loops | xs:long | total_hash_loops - Total Hash Loops monitor element |
| total_implicit_compilations | xs:long | total_implicit_compilations - Total implicit complications |
| total_implicit_compile_proc_time | xs:long | total_implicit_compile_proc_time - Total implicit compile processing time |
| total_implicit_compile_time | xs:long | total_implicit_compile_time - Total implicit compile time |
| total_index_build_proc_time | xs:long | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation |
| total_index_build_time | xs:long | total_index_build_time - Total time spent building indexes due to index creation |
| total_indexes_built | xs:long | total_indexes_built - Total number of indexes built |
| total_load_proc_time | xs:long | total_load_proc_time - Total load processing time |
| total_load_time | xs:long | total_load_time - Total load time |
| total_loads | xs:long | total_loads - Total loads |
| total_nested_invocations | xs:long | total_nested_invocations - Total nested invocations monitor element |
| total_olap_funcs | xs:long | total_olap_funcs - Total OLAP Functions monitor element |

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|-----------------------------------|-----------|---|
| total_peas | xs:long | total_peas - Total partial early aggregations |
| total_peds | xs:long | total_peds - Total partial early distincts |
| total_reorg_proc_time | xs:long | total_reorg_proc_time - Total reorganization processing time |
| total_reorg_time | xs:long | total_reorg_time - Total reorganization time |
| total_reorgs | xs:long | total_reorgs - Total reorganizations |
| total_rollback_proc_time | xs:long | total_rollback_proc_time - Total rollback processing time |
| total_rollback_time | xs:long | total_rollback_time - Total rollback time |
| total_routine_coord_time | xs:long | total_routine_coord_time - Total routine coordinator time monitor element |
| total_routine_invocations | xs:long | total_routine_invocations - Total routine invocations. The total number of times a routine was invoked by this routine. |
| total_routine_time | xs:long | total_routine_time - Total routine time. |
| total_routine_user_code_time | xs:long | total_routine_user_code_time - Total routine user code time |
| total_routine_user_code_proc_time | xs:long | total_routine_user_code_proc_time - Total routine user code processing time |
| total_rqst_time | xs:long | total_rqst_time - Total request time. |
| total_runstats | xs:long | total_runstats - Total runtime statistics |
| total_runstats_proc_time | xs:long | total_runstats_proc_time - Total runtime statistics processing time |
| total_runstats_time | xs:long | total_runstats_time - Total runtime statistics |
| total_section_proc_time | xs:long | total_section_proc_time - Total section processing time |
| total_section_sort_proc_time | xs:long | total_section_sort_proc_time - Total section sort processing time |
| total_section_sort_time | xs:long | total_section_sort_time - Total section sort time |
| total_section_sorts | xs:long | total_section_sorts - Total section sorts |
| total_section_time | xs:long | total_section_time - Total section time |
| total_sorts | xs:long | total_sorts - Total sorts |
| total_stats_fabrication_proc_time | xs:long | total_stats_fabrication_proc_time - Total statistics fabrication processing time |
| total_stats_fabrication_time | xs:long | total_stats_fabrication_time - Total statistics fabrication time |
| total_stats_fabrications | xs:long | total_stats_fabrications - Total statistics fabrications |
| total_sync_runstats_proc_time | xs:long | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time |
| total_sync_runstats | xs:long | total_sync_runstats - Total synchronous RUNSTATS activities |

Table 149. Detailed metrics returned for MON_GET_ROUTINE_DETAILS (continued)

| Element name | Data type | Description or corresponding monitor element |
|-----------------------------|-----------|---|
| total_sync_runstats_time | xs:long | total_sync_runstats_time - Total synchronous RUNSTATS time |
| total_times_routine_invoked | xs:long | total_times_routine_invoked - Total routine invoked occurrences monitor element |
| total_wait_time | xs:long | total_wait_time - Total wait time |
| tq_sort_heap_rejections | xs:long | tq_sort_heap_rejections - Table queue sort heap rejections |
| tq_sort_heap_requests | xs:long | tq_sort_heap_requests - Table queue sort heap requests |
| tq_tot_send_spills | xs:long | tq_tot_send_spills - Total number of table queue buffers overflowed |
| uid_sql_stmts | xs:long | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| wlm_queue_assignments_total | xs:long | wlm_queue_assignments_total - Workload manager total queue assignments |
| wlm_queue_time_total | xs:long | wlm_queue_time_total - Workload manager total queue time |
| xquery_stmts | xs:long | xquery_stmts - XQuery Statements Attempted monitor element |

Usage notes

Use the MON_GET_ROUTINE_DETAILS table function to identify the most expensive routines on the database server.

This table function returns one row of metrics for each routine or trigger and each member that matches the input arguments. Input argument values are complementary. Metrics returned are aggregates of all executions of the routine on that member. No aggregation across members is performed. However, an aggregation across members is possible through SQL queries (as shown in the Examples section). If a routine executes subroutines, the work done in the subroutines is included in the metrics of the parent routine.

Routine monitoring data collection must be explicitly enabled using the mon_rtn_data database configuration parameter. If this configuration parameter is set to NONE, no information is returned.

The counters and time-spent monitor elements returned by this table function are controlled with the COLLECT REQUEST METRICS clause on service classes and the mon_req_metrics database configuration parameter at the database level. If neither control is enabled, the counters and time-spent monitor elements reported are 0.

When the package for a dynamically prepared compound SQL statement has been removed from the package cache, information for this routine will no longer be reported by MON_GET_ROUTINE_DETAILS function. Similarly, when a routine is dropped, information about the routine is no longer be reported.

Any routines that were not executed during the previous 24 hour period are pruned from memory and not returned.

The schema for the XML document that is returned in the DETAILS column is available in the file `sqllib/misc/DB2MonRoutines.xsd`. Further details can be found in the file `sqllib/misc/DB2MonCommon.xsd`.

Example

List where time was spent during the processing of the TEST.PROC1 stored procedure, identifying the top 5 areas where time is spent.

```
SELECT B.* FROM TABLE(MON_GET_ROUTINE_DETAILS('P','TEST',NULL,'PROC1', -1))
AS A, TABLE(MON_FORMAT_XML_TIMES_BY_ROW(A.DETAILS)) AS B
ORDER BY TOTAL_TIME_VALUE DESC FETCH FIRST 5 ROWS ONLY
```

returns

| METRIC_NAME | TOTAL_TIME_VALUE | COUNT | PARENT_METRIC_NAME |
|-----------------------------------|------------------|-------|--------------------|
| TOTAL_RQST_TIME | 1055 | 0 | - |
| TOTAL_ROUTINE_USER_CODE_PROC_TIME | 673 | 0 | TOTAL_RQST_TIME |
| TOTAL_WAIT_TIME | 317 | - | TOTAL_RQST_TIME |
| POOL_READ_TIME | 236 | 142 | TOTAL_WAIT_TIME |
| DIRECT_READ_TIME | 81 | 39 | TOTAL_WAIT_TIME |

5 record(s) selected.

MON_GET_ROUTINE_EXEC_LIST table function - get list of statements executed by routine

The MON_GET_ROUTINE_EXEC_LIST table function returns a list of all statements (sections) executed by each procedure, external function, compiled function, compiled trigger, and anonymous block invoked since the database was activated.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- SQLADM authority
- DBADM authority

Default PUBLIC privilege

None

Syntax

```
►►MON_GET_ROUTINE_EXEC_LIST(—routine_type—,—routine_schema—,—►
►—routine_module_name—,—routine_name—,—member—)►►
```

The schema is SYSPROC.

Routine parameters

routine_type

An input parameter of type CHAR(2) that specifies the type of routine or compiled trigger for which data is be returned:

- "P" for a procedure
- "SP" for the specific name of a procedure
- "F" for a compiled function
- "SF" for the specific name of a compiled function
- "T" for a compiled trigger
- "C" for a dynamically prepared compound SQL statement in SQL PL or an anonymous block in PL/SQL

If the "P", "F", "T", or "C" routine type is specified and no input name is specified, all routines of the given type are returned. If the routine type is "SP" or "SF", the specific name of the routine must be provided.

If the routine type is an empty string, NULL, or blanks, all routines of all types are returned.

routine_schema

An input parameter of type VARCHAR(128) that specifies the schema for the routine or trigger. For dynamically prepared compound SQL statements or anonymous blocks, the schema can be determined using the MON_GET_SECTION_ROUTINES table function. Use NULL or an empty string to return the routines and triggers in all schemas. This parameter is case sensitive.

routine_module_name

An input parameter of type VARCHAR(128) that specifies the name of the module for the input routine, if applicable. Using NULL or an empty string to return the routines in all modules. This parameter is case sensitive.

routine_name

An input parameter of type VARCHAR(128) that specifies the name of the routine. If the input parameter is "SP" or "SF", the specific name of the routine must be provided. For dynamically prepared compound SQL statements or anonymous blocks, the name can be determined using the MON_GET_SECTION_ROUTINES table function. Using NULL or an empty string to return all routines matching the other input parameters. This parameter is case sensitive.

member

An input parameter of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is implicitly set.

Information returned

Table 150. Information returned for MON_GET_ROUTINE_EXEC_LIST

| Column name | Data type | Description or corresponding monitor element |
|----------------|--------------|---|
| ROUTINE_TYPE | CHAR(1) | routine_type - Routine type monitor element |
| ROUTINE_SCHEMA | VARCHAR(128) | routine_schema - Routine schema monitor element |

Table 150. Information returned for MON_GET_ROUTINE_EXEC_LIST (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------------|--------------------------|---|
| ROUTINE_MODULE_NAME | VARCHAR(128) | routine_module_name - Routine module name monitor element |
| ROUTINE_NAME | VARCHAR(128) | routine_name - Routine name monitor element |
| SPECIFIC_NAME | VARCHAR(128) | specific_name - Specific name monitor element |
| DYN_COMPOUND_EXEC_ID | VARCHAR(32) FOR BIT DATA | dyn_compound_exec_id - Dynamic compound statement executable identifier monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| ROUTINE_ID | INTEGER | routine_id - Routine ID monitor element |
| SUBROUTINE_ID | INTEGER | subroutine_id - Subroutine identifier monitor element |
| LIB_ID | BIGINT | lib_id - Library identifier monitor element |
| SECTION_TYPE | CHAR(1) | section_type - Section type indicator. |
| PACKAGE_SCHEMA | VARCHAR(128) | package_schema - Package schema. |
| PACKAGE_NAME | VARCHAR(128) | package_name - Package name. |
| PACKAGE_VERSION_ID | VARCHAR(64) | package_version_id - Package version ID. |
| SECTION_NUMBER | BIGINT | section_number - Section number. |
| STMTNO | INTEGER | stmtno - Statement number monitor element |
| EXECUTABLE_ID | VARCHAR(32) FOR BIT DATA | executable_id - Executable ID. |
| NUM_ROUTINES | INTEGER | num_routines - Number of routines |
| CALL_STMT_ROUTINE_ID | INTEGER | call_stmt_routine_id - Call statement routine identifier monitor element |
| CALL_STMT_SUBROUTINE_ID | INTEGER | call_stmt_subroutine_id - Call statement subroutine identifier monitor element |
| NUM_EXECUTIONS | BIGINT | num_executions - Statement executions. |
| NUM_EXEC_WITH_METRICS | BIGINT | num_exec_with_metrics - Number of executions with metrics collected. |
| NUM_COORD_EXEC | BIGINT | num_coord_exec - Number of executions by coordinator agent. |
| NUM_COORD_EXEC_WITH_METRICS | BIGINT | num_coord_exec_with_metrics - Number of executions by coordinator agent with metrics. |
| COORD_STMT_EXEC_TIME | BIGINT | coord_stmt_exec_time - Execution time for statement by coordinator agent |
| TOTAL_ACT_TIME | BIGINT | total_act_time - Total activity time |
| TOTAL_ACT_WAIT_TIME | BIGINT | total_act_wait_time - Total activity wait time |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |

Table 150. Information returned for MON_GET_ROUTINE_EXEC_LIST (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post threshold sorts |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post shared threshold sorts |
| ROWS_READ | BIGINT | rows_read - Rows read |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time |
| TOTAL_SORTS | BIGINT | total_sorts - Total sorts |

Usage notes

The MON_GET_ROUTINE_EXEC_LIST table function returns one row for each unique statement (section) executed by a routine or trigger. The function also returns a set of metrics aggregated across all executions of that statement within the routine or trigger. No aggregation across members is performed. However, an aggregation across members is possible through SQL queries (as shown in the Examples section).

The aggregate metrics do not include metrics for any child statements. For example, if a CALL statement is executed by a routine, the metrics returned in MON_GET_ROUTINE_EXEC_LIST for the CALL statement do not include metrics for any work executed by other statements invoked by the CALL. The only exception is coord_stmt_exec_time monitor element that returns the elapsed time of the statement and so implicitly captures any time spent in child statements.

Routine monitoring data collection and statement monitoring must be explicitly enabled using the mon_rtn_data and mon_rtn_execlist database configuration parameters. If these configuration parameters are disabled, no information is returned.

The counters and time-spent monitor elements returned by this table function are controlled with the COLLECT ACTIVITY METRICS clause on workloads and the mon_act_metrics database configuration parameter at the database level. If neither control is enabled, the counters and time-spent monitor elements reported are 0.

When the package for a dynamically prepared compound SQL statement or an anonymous block is cleaned up from the package cache, information for this statement is no longer be reported by MON_GET_ROUTINE_EXEC_LIST function. Similarly, when a routine or trigger is dropped, information about the routine or trigger is no longer be reported.

Statements executed by routines that were subsequently dropped or not executed during the previous 24 hour period are pruned from memory and not returned.

Examples

1. List all the statements executed by function TEST.X.

```
SELECT SUBSTR(P.STMT_TEXT,1,45) AS TEXT
FROM TABLE(MON_GET_ROUTINE_EXEC_LIST('F', 'TEST', NULL, 'X', -1))
AS T, TABLE(MON_GET_PKG_CACHE_STMT(NULL,NULL,-1)) AS P
WHERE T.EXECUTABLE_ID = P.EXECUTABLE_ID
```

returns

```
TEXT
-----
WITH PID_VALUES (WIDEBAND,ENGN_TEMP) AS SEL
insert into GSO_LOGS values(8.7, 145, 1.406)
call SYSIBMSUBROUTINE.TEST_66613_1157394573()
```

3 record(s) selected.

2. List all the statements in procedure TEST.PROC1 by the percentage of coordinator statement execution time that they are consuming.

```
SELECT 100*B.COORD_STMT_EXEC_TIME / A.TOTAL_ROUTINE_COORD_EXEC_TIME
AS PERCENT_EXEC_TIME, SUBSTR(C.STMT_TEXT,1,45)
AS STMT_TEXT FROM TABLE(MON_GET_ROUTINE('SP',NULL,NULL,'PROC1', -1)) AS A,
TABLE(MON_GET_ROUTINE_EXEC_LIST('SP',NULL,NULL,'PROC1', -1)) AS B,
TABLE(MON_GET_PKG_CACHE_STMT(NULL,NULL,NULL,-1)) AS C
WHERE B.EXECUTABLE_ID = C.EXECUTABLE_ID ORDER BY B.COORD_STMT_EXEC_TIME DESC
```

returns

```
PERCENT_EXEC_TIME STMT_TEXT
-----
10 SELECT WAFR, MPHX64, ENGN_RPM FROM KB28_LOGS
3 SELECT ( ENGN_TQ * ENGN_RPM )/5252 AS HP FRO
```

2 record(s) selected.

MON_GET_RTS_RQST table function - Retrieve information about real-time statistics requests

The MON_GET_RTS_RQST table function returns information about all real-time statistics requests that are pending in the system and the set of requests that are currently being processed by the real-time statistics daemon, such as on the real-time statistics processing queue.

Syntax

►►—MON_GET_RTS_RQST—()—————►►

The schema is SYSPROC.

Authorization

You require one of the following types of authorization to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

The following query displays all pending and currently queued or executing real-time statistics requests:

```

SELECT MEMBER,
       QUEUE_POSITION,
       REQUEST_STATUS,
       REQUEST_TYPE,
       OBJECT_TYPE,
       VARCHAR(OBJECT_SCHEMA, 10) AS SCHEMA,
       VARCHAR(OBJECT_NAME, 10) AS NAME
FROM TABLE(MON_GET_RTS_RQST()) AS T
ORDER BY MEMBER, QUEUE_POSITION ASC

```

Following is an example of output from this query:

| MEMBER | QUEUE_POSITION | REQUEST_STATUS | REQUEST_TYPE | OBJECT_TYPE | SCHEMA | NAME |
|--------|----------------|----------------|---------------|-------------|--------|----------|
| 0 | 1 | EXECUTING | COLLECT_STATS | TABLE | TEST | EMPLOYEE |
| 0 | 2 | QUEUED | COLLECT_STATS | TABLE | TEST | T1 |
| 0 | 3 | QUEUED | WRITE_STATS | TABLE | TEST | T3 |
| 0 | - | PENDING | WRITE_STATS | TABLE | TEST | BLAH |
| 1 | - | PENDING | COLLECT_STATS | TABLE | TEST | DEPT |
| 1 | - | PENDING | WRITE_STATS | TABLE | TEST | SALES |
| 2 | - | PENDING | WRITE_STATS | TABLE | TEST | SALARY |

7 record(s) selected.

Usage notes

The MON_GET_RTS_RQST table function returns information about all real-time statistics requests that are pending on all members. The queue for processing real-time statistics requests exists only on a single member.

The MON_GET_RTS_RQST table function does not report any real-time statistics requests if real-time statistics collection is not enabled.

Information returned

Table 151. Information returned for MON_GET_RTS_RQST function

| Column name | Data type | Description |
|---------------|--------------|--|
| OBJECT_TYPE | VARCHAR(8) | Type of object, which is one of the following values: <ul style="list-style-type: none"> TABLE VIEW |
| OBJECT_SCHEMA | VARCHAR(128) | object_schema - Object schema monitor element |
| OBJECT_NAME | VARCHAR(128) | object_name - Object name monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| REQUEST_TYPE | VARCHAR(14) | One of the following values: <p>WRITE_STATS
Request is to write already collected statistics to the catalogs.</p> <p>COLLECT_STATS
Request is to perform full statistics collection.</p> |

Table 151. Information returned for MON_GET_RTS_RQST function (continued)

| Column name | Data type | Description |
|----------------------|-------------|--|
| REQUEST_STATUS | VARCHAR(10) | One of the following values:

PENDING
Request is waiting to be picked up by the real-time statistics daemon.

QUEUED
Request was gathered by the real-time statistics daemon and is awaiting processing.

EXECUTING
Request is currently being processed by the real-time statistics daemon. |
| REQUEST_TIME | TIMESTAMP | Time that the request was submitted. |
| QUEUE_POSITION | INTEGER | If the value of the REQUEST_STATUS column is QUEUED, the position of the request in the real-time statistics daemon processing queue. |
| QUEUE_ENTRY_TIME | TIMESTAMP | If the value of the REQUEST_STATUS column is QUEUED, the time that the request was added to the real-times statistics daemon processing queue. |
| EXECUTION_START_TIME | TIMESTAMP | If the value of the REQUEST_STATUS column is EXECUTION, the time that the request began execution; NULL otherwise. |

MON_GET_SECTION table function - Get a copy of a section from the package cache

The MON_GET_SECTION table function returns a copy of a section for a dynamic or static SQL statement from the package cache. The section that is returned can be saved for future analysis. For example, when you investigate a plan change for a statement, the saved section can be passed as input to the EXPLAIN_FROM_DATA stored procedure to examine the previous access plan.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- SQLADM authority
- DBADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_GET_SECTION—(—executable_id—)—————►►

The schema is SYSPROC.

Routine parameters

executable_id

An input argument of type VARCHAR (32) FOR BIT DATA that specifies a unique section in the database package cache.

The routine searches the package cache on all members for the section. For static SQL, the catalogs are searched if the section is no longer in the package cache.

If a null or invalid value is specified, SQL0171 is returned.

Information returned

Table 152. Information returned by the MON_GET_SECTION

| Column name | Data type | Description |
|--------------|--------------|---|
| SECTION_TYPE | CHAR(1) | section_type - Section type indicator monitor element |
| SECTION_ENV | BLOB(134 MB) | section_env - Section environment monitor element |

Example

Use the following query to retrieve the statement text, statement identifier, plan identifier, and section information from the compilation environment. Information is returned for all statements currently in the package cache. The information is inserted into a table called REPOSITORY_SECTIONS.

```
INSERT INTO
    REPOSITORY_SECTIONS(STMT_TEXT, STMTID, PLANID,
                        SEMANTIC_ENV_ID, SECTION_DATA)
SELECT STMT_TEXT, STMTID, PLANID, SEMANTIC_ENV_ID,
( SELECT B.SECTION_ENV
  FROM TABLE(MON_GET_SECTION(A.EXECUTABLE_ID)) AS B)
SECTION_DATA
FROM TABLE(MON_GET_PKG_CACHE_STMT(NULL,NULL,NULL,-2)) AS A
```

You can use this information to build a history of statements that ran. You can use the EXPLAIN_FROM_DATA stored procedure to examine the access plan for each saved statement by passing the saved section to the stored procedure.

MON_GET_SECTION_OBJECT table function - List objects accessed by a section

The MON_GET_SECTION_OBJECT table function retrieves the list of objects that are accessed by a section. This information can be used in filtering or for later analysis.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- SQLADM authority
- DBADM authority

Default PUBLIC privilege

None

Syntax

```
►►—MON_GET_SECTION_OBJECT—(—executable_id—,—object_type—,——————►  
►—object_schema—,—object_module—,—object_name—,—member—)——————►◄
```

The schema is SYSPROC.

Routine parameters

executable_id

An optional input parameter of type VARCHAR(32) FOR BIT DATA that uniquely identifies a section for which the list of objects is to be returned. If this argument is NULL then information for all sections in the package cache are returned.

object_type

An optional input parameter of type VARCHAR(16) that identifies the object types to be returned. If this argument is null or an empty string, then all types of objects are returned. Valid object types are listed in Table 153.

object_schema

An optional input parameter of type VARCHAR(128) that identifies the schema for the objects to be returned. If this argument is null or an empty string, then objects in all schemas are returned.

object_module

An optional input parameter of type VARCHAR(128) that identifies the module of the objects to be returned. If this argument is null or an empty string, then objects defined in all modules are returned.

object_name

An optional input parameter of type VARCHAR(128) that identifies the name of the object to be returned. If this argument is null or an empty string, then objects are not filtered by name.

member

An optional input argument of type SMALLINT that specifies a valid member in the same instance as the currently connected database when this function is called. Specify **-1** for the current database member or **-2** for all active database members. If the null value is specified, **-1** is set.

Table 153. Possible values for the OBJECT_TYPE parameter

| Value | Description |
|-------|---|
| C | Dynamically prepared compound SQL statement or PL/SQL anonymous block |
| CO | Column-organized table |
| DP | Data partitioned table |
| F | Function |
| IX | Index |
| LI | Partitioned index |
| LP | Partitioned physical XML index |

Table 153. Possible values for the OBJECT_TYPE parameter (continued)

| Value | Description |
|-------|-------------------------------|
| LX | Partitioned logical XML index |
| NK | Nickname |
| P | Procedure |
| PI | Physical XML index |
| RX | RCT index |
| T | Trigger |
| TA | Table |
| XI | Logical XML index |

Information returned

Table 154. Information returned by the MON_GET_SECTION_OBJECT

| Column name | Data type | Description |
|---------------|--------------------------|---|
| MEMBER | SMALLINT | member - Database member monitor element |
| EXECUTABLE_ID | VARCHAR(32) FOR BIT DATA | executable_id - Executable ID monitor element |
| OBJECT_TYPE | VARCHAR(16) | An object type, with the following valid values: <ul style="list-style-type: none"> • C - Dynamically prepared compound SQL statement or PL/SQL anonymous block • CO - Column-organized table • DP - Data partitioned table • F - Function • IX - Index • LI - Partitioned index • LP - Partitioned physical XML index • LX - Partitioned logical XML index • NK - Nickname • P - Procedure • PI - Physical XML index • RX - RCT index • T - Trigger • TA - Table • XI - Logical XML index |
| OBJECT_SCHEMA | VARCHAR(128) | object_schema - Object schema monitor element |
| OBJECT_MODULE | VARCHAR(128) | object_module - Object module monitor element |
| OBJECT_NAME | VARCHAR(128) | object_name - Object name monitor element |

[illegible]

Sample output from this query is as follows:

3 record(s) selected.

The MON_GET_SECTION_ROUTINE table function returns a list of all procedures, external functions, compiled functions, and compiled triggers that might be invoked during the execution of the input section.

One of the following authorizations is required:

- ## Default PUBLIC privilege

Syntax

The schema is SYSPROC.

executable id

758 Db2 11.1 for Linux, UNIX, and Windows: Built-In Routines and Views

Information returned

Table 155. Information returned for MON_GET_SECTION_ROUTINE

| Column name | Data type | Description or corresponding monitor element |
|---------------------|--------------|---|
| ROUTINE_TYPE | CHAR(1) | routine_type - Routine type monitor element |
| ROUTINE_SCHEMA | VARCHAR(128) | routine_schema - Routine schema monitor element |
| ROUTINE_MODULE_NAME | VARCHAR(128) | routine_module_name - Routine module name monitor element |
| ROUTINE_NAME | VARCHAR(128) | routine_name - Routine name monitor element |
| SPECIFIC_NAME | VARCHAR(128) | specific_name - Specific name monitor element |
| ROUTINE_ID | INTEGER | routine_id - Routine ID monitor element |
| SUBROUTINE_ID | INTEGER | subroutine_id - Subroutine identifier monitor element |
| LIB_ID | BIGINT | lib_id - Library identifier monitor element |

Usage notes

The input section must reside in the package cache.

Objects returned by this function are not necessarily invoked during execution of the input section. For example, in the following select statement FUNCTION1 is a compiled function:

```
SELECT CASE WHEN COL1 = 0 THEN FUNCTION1() ELSE NULL END AS T1 FROM TABLE1
```

The MON_GET_SECTION_ROUTINE table function will return a row with the FUNCTION1 function when executable_id is passed as an argument, identifying the section for this statement. However, FUNCTION1 will be invoked during execution of the statement only if TABLE1 contains the value 0 for COL1 in one of its rows.

Examples

- List all routines that might be executed by the section in the package cache with executable ID
x'010000000000000052010000000000001000000010020120822205618607103'.

```
SELECT * FROM
  TABLE(MON_GET_SECTION_ROUTINE(
    x'010000000000000052010000000000001000000010020120822205618607103'))
  AS T
```

returns

```
ROUTINE_TYPE ROUTINE_SCHEMA ROUTINE_MODULE_NAME ROUTINE_NAME ...
-----
          F SPEEDY          TANKS          ANALYZE_LOG1 ...
          P SPEEDY          TANKS          POST_PROCESS ...
```

2 record(s) selected.

- A user notices that the insert statement with executable ID `x'010000000000000052010000000000001000000010020120822205618607103'` ran for a long time. Run the following statement to check if any compiled triggers were invoked during the execution of the insert:

```
SELECT * FROM
  TABLE(MON_GET_SECTION_ROUTINE(
    x'010000000000000052010000000000001000000010020120822205618607103'))
  AS T WHERE ROUTINETYPE = 'T'
```

returns

| ROUTINE_TYPE | ROUTINE_SCHEMA | ROUTINE_MODULE_NAME | ROUTINE_NAME | ... |
|--------------|----------------|---------------------|--------------------|-----|
| T | MIKE | | INVALID_WELD_VALUE | ... |
| P | MIKE | | INSERT_WELD_SPECS | ... |

2 record(s) selected.

- Find the internal schema and procedure name for an anonymous block with executable ID

`x'010000000000000052010000000000001000000010020120822205618607103'`.

```
SELECT ROUTINESCHEMA, ROUTINENAME, SPECIFICNAME
  FROM TABLE(MON_GET_SECTION_ROUTINE(
    x'010000000000000052010000000000001000000010020120822205618607103'))
  AS T
```

returns

| ROUTINE_SCHEMA | ROUTINE_NAME | SPECIFIC_NAME | ... |
|----------------|--------------|--------------------|-----|
| GSOSchema | PING RUBBLE | SQL120801135351900 | ... |

1 record(s) selected.

MON_GET_SERVERLIST table function - get member priority details

The MON_GET_SERVERLIST table function returns metrics on the server list for the currently connected database as cached on one or more members.

For each active member, the server list contains its connectivity and priority information, which enables a remote client to perform workload balancing (WLB) and automatic client reroute (ACR). The member parameter corresponds to the member where the server list is cached.

Syntax

```
►► MON_GET_SERVERLIST ( (member [member_subset_name]) ) ◀◀
```

The schema is SYSPROC.

Table function parameters

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. That is, -1 is equivalent to issuing `db2pd -serverlist -db dbname`, where `dbname` is the name of the database currently connected to, only on the current member, whereas -2 is equivalent to issuing the same command for all database members. If the NULL value is specified, -1 is set implicitly.

member_subset_name

An input argument of type VARCHAR(128) that identifies the member subset for which server list metrics should be returned. If the argument is null or the empty string, metrics are returned for all server lists. The default value is NULL.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Information returned

Table 156. Information returned for MON_GET_SERVERLIST

| Column name | Data type | Description |
|--------------------------|--------------|--|
| MEMBER | SMALLINT | member - Database member monitor element |
| CACHED_TIMESTAMP | TIMESTAMP | cached_timestamp - Cached timestamp |
| HOSTNAME | VARCHAR(255) | hostname - Host name monitor element |
| PORT_NUMBER | INTEGER | port_number - Port number |
| SSL_PORT_NUMBER | INTEGER | ssl_port_number - SSL port number |
| PRIORITY | SMALLINT | priority - Priority value |
| MEMBER_SUBSET_ID | INTEGER | member_subset_id - Member subset ID |
| SERVER_LIST_ENTRY_MEMBER | SMALLINT | server_list_entry_member - Member ID for the member in the server list |

Usage notes

- The MON_GET_SERVERLIST table function returns one row for each member address cached in a server list that matches the input arguments. Each row contains connectivity information and relative priority which enable a remote client to perform workload balancing and automatic client reroute.
- The MEMBER column indicates which member cached the server list information; the SUBSET_ID column indicates what member subset the server list information was cached for.

Examples

1. Assume a connection to database SAMPLE on member 0, where the database has been accessed by a remote client. List all server list metrics as cached on this member for this database:

```
SELECT MEMBER, CACHED_TIMESTAMP, VARCHAR(HOSTNAME, 30)
  AS HOSTNAME, PORT_NUMBER, SSL_PORT_NUMBER, PRIORITY
FROM TABLE (MON_GET_SERVERLIST (-1))
```

Output results for connection to database SAMPLE on member 0.

| MEMBER | CACHED_TIMESTAMP | HOSTNAME | ... |
|--------|----------------------------|----------------------------|-----|
| 0 | 2011-02-19-17.39.33.000000 | coralxib23.torolab.ibm.com | ... |
| 0 | 2011-02-19-17.39.33.000000 | coralxib24.torolab.ibm.com | ... |

Output for query (continued).

| ... | PORT_NUMBER | SSL_PORT_NUMBER | PRIORITY |
|-----|-------------|-----------------|----------|
| ... | 49712 | 0 | 67 |
| ... | 49712 | 0 | 32 |

2. Assume a connection to database SAMPLE on member 0 and that the database has been accessed on both members 0 and 1 by a remote client. List all server list metrics as cached on both members for this database:

```
SELECT MEMBER, CACHED_TIMESTAMP, VARCHAR(HOSTNAME, 30)
AS HOSTNAME, PORT_NUMBER, SSL_PORT_NUMBER, PRIORITY
FROM TABLE (MON_GET_SERVERLIST (-2))
```

Output results for connection to database SAMPLE on member 0, which is currently active on members 0 and 1.

| MEMBER | CACHED_TIMESTAMP | HOSTNAME | ... |
|--------|----------------------------|----------------------------|-----|
| 0 | 2011-02-19-17.39.33.000000 | coralxib23.torolab.ibm.com | ... |
| 0 | 2011-02-19-17.39.33.000000 | coralxib24.torolab.ibm.com | ... |
| 1 | 2011-02-19-17.39.33.000000 | coralxib24.torolab.ibm.com | ... |
| 1 | 2011-02-19-17.39.33.000000 | coralxib23.torolab.ibm.com | ... |

Output for query (continued).

| ... | PORT_NUMBER | SSL_PORT_NUMBER | PRIORITY |
|-----|-------------|-----------------|----------|
| ... | 49712 | 0 | 67 |
| ... | 49712 | 0 | 32 |
| ... | 49712 | 0 | 32 |
| ... | 49712 | 0 | 67 |

3. Assume a connection to the database alias MY_ALIAS exists on member 0 and the database has been accessed on all members. Assume that the connection has been assigned to the member subset MY_SUBSET which includes members 0 and 1 and excludes member 2. List the server list metrics for the member subset MY_SUBSET as cached on member 0.

```
SELECT MEMBER, SUBSET_ID, CACHED_TIMESTAMP, VARCHAR(HOSTNAME, 30)
AS HOSTNAME, PORT_NUMBER, SSL_PORT_NUMBER, PRIORITY
FROM TABLE (MON_GET_SERVERLIST(0, 'MY_SUBSET'))
```

This query returns the following output:

| MEMBER | SUBSET_ID | CACHED_TIMESTAMP | HOSTNAME | PORT_NUMBER |
|--------|-----------|----------------------------|----------------------------|-------------|
| 0 | 1 | 2011-02-19-17.39.33.000000 | coralxib14.torolab.ibm.com | 33163 |
| 0 | 1 | 2011-02-19-17.39.33.000000 | coralxib15.torolab.ibm.com | 33163 |

MON_GET_SERVICE_SUBCLASS table function - Get service subclass metrics

The MON_GET_SERVICE_SUBCLASS table function returns metrics for one or more service subclasses.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority

- SQLADM authority

Default PUBLIC privilege

None

Syntax

```
►►—MON_GET_SERVICE_SUBCLASS—(—service_superclass_name—,—————►
►—service_subclass_name—,—member—)—————►◄
```

The schema is SYSPROC.

Routine parameters

service_superclass_name

An input argument of type VARCHAR(128) that specifies a valid service superclass name in the currently connected database when calling this function. If the argument is null or an empty string, metrics are retrieved for all the superclasses in the database.

service_subclass_name

An input argument of type VARCHAR(128) that specifies a valid service subclass name in the currently connected database when calling this function. If the argument is null or an empty string, metrics are retrieved for all the subclasses in the database.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

Information returned

Note: In the rows returned by this function that describe the default subclass SYSDEFAULTSUBCLASS under the superclass SYSDEFAULTSYSTEMCLASS, all columns that report metrics have a value of 0.

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS

| Column Name | Data Type | Description |
|-------------------------|--------------|---|
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name monitor element |
| SERVICE_SUBCLASS_NAME | VARCHAR(128) | service_subclass_name - Service subclass name monitor element |
| SERVICE_CLASS_ID | INTEGER | service_class_id - Service class ID monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| ACT_ABORTED_TOTAL | BIGINT | act_aborted_total - Total aborted activities monitor element |
| ACT_COMPLETED_TOTAL | BIGINT | act_completed_total - Total completed activities monitor element |
| ACT_REJECTED_TOTAL | BIGINT | act_rejected_total - Total rejected activities monitor element |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|-------------------------|-----------|---|
| AGENT_WAIT_TIME | BIGINT | agent_wait_time - Agent wait time monitor element |
| AGENT_WAITS_TOTAL | BIGINT | agent_waits_total - Total agent waits monitor element |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads monitor element |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads monitor element |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads monitor element |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer pool XDA data logical reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads monitor element |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads monitor element |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads monitor element |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads monitor element |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer pool XDA data physical reads monitor element |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes monitor element |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes monitor element |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer pool XDA data writes monitor element |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time monitor element |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time monitor element |
| CLIENT_IDLE_WAIT_TIME | BIGINT | client_idle_wait_time - Client idle wait time monitor element |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected monitor element |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database monitor element |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database monitor element |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time monitor element |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|-----------------------|-----------|---|
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests monitor element |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests monitor element |
| FCM_RECV_VOLUME | BIGINT | fcm_recv_volume - FCM received volume monitor element |
| FCM_RECVS_TOTAL | BIGINT | fcm_recvs_total - FCM receives total monitor element |
| FCM_SEND_VOLUME | BIGINT | fcm_send_volume - FCM send volume monitor element |
| FCM_SENDS_TOTAL | BIGINT | fcm_sends_total - FCM sends total monitor element |
| FCM_RECV_WAIT_TIME | BIGINT | fcm_recv_wait_time - FCM received wait time monitor element |
| FCM_SEND_WAIT_TIME | BIGINT | fcm_send_wait_time - FCM send wait time monitor element |
| IPC_RECV_VOLUME | BIGINT | ipc_recv_volume - Interprocess communication received volume monitor element |
| IPC_RECV_WAIT_TIME | BIGINT | ipc_recv_wait_time - Interprocess communication received wait time monitor element |
| IPC_RECVS_TOTAL | BIGINT | ipc_recvs_total - Interprocess communication receives total monitor element |
| IPC_SEND_VOLUME | BIGINT | ipc_send_volume - Interprocess communication send volume monitor element |
| IPC_SEND_WAIT_TIME | BIGINT | ipc_send_wait_time - Interprocess communication send wait time monitor element |
| IPC_SENDS_TOTAL | BIGINT | ipc_sends_total - Interprocess communication send total monitor element |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations monitor element |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts monitor element |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks monitor element |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits monitor element |
| LOG_BUFFER_WAIT_TIME | BIGINT | log_buffer_wait_time - Log buffer wait time monitor element |
| NUM_LOG_BUFFER_FULL | BIGINT | num_log_buffer_full - Number of times full log buffer caused agents to wait monitor element |
| LOG_DISK_WAIT_TIME | BIGINT | log_disk_wait_time - Log disk wait time monitor element |
| LOG_DISK_WAITS_TOTAL | BIGINT | log_disk_waits_total - Total log disk waits monitor element |
| RQSTS_COMPLETED_TOTAL | BIGINT | rqsts_completed_total - Total requests completed monitor element |
| ROWS_MODIFIED | BIGINT | rows_modified - Rows modified monitor element |
| ROWS_READ | BIGINT | rows_read - Rows read monitor element |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|------------------------------|-----------|--|
| ROWS_RETURNED | BIGINT | rows_returned - Rows returned monitor element |
| TCPIP_RECV_VOLUME | BIGINT | tcPIP_recv_volume - TCP/IP received volume monitor element |
| TCPIP_SEND_VOLUME | BIGINT | tcPIP_send_volume - TCP/IP send volume monitor element |
| TCPIP_RECV_WAIT_TIME | BIGINT | tcPIP_recv_wait_time - TCP/IP received wait time monitor element |
| TCPIP_RECVS_TOTAL | BIGINT | tcPIP_recv_total - TCP/IP receives total monitor element |
| TCPIP_SEND_WAIT_TIME | BIGINT | tcPIP_send_wait_time - TCP/IP send wait time monitor element |
| TCPIP_SENDS_TOTAL | BIGINT | tcPIP_sends_total - TCP/IP sends total monitor element |
| TOTAL_APP_RQST_TIME | BIGINT | total_app_rqst_time - Total application request time monitor element |
| TOTAL_RQST_TIME | BIGINT | total_rqst_time - Total request time monitor element |
| WLM_QUEUE_TIME_TOTAL | BIGINT | wlm_queue_time_total - Workload manager total queue time monitor element |
| WLM_QUEUE_ASSIGNMENTS_TOTAL | BIGINT | wlm_queue_assignments_total - Workload manager total queue assignments monitor element |
| TOTAL_RQST_MAPPED_IN | BIGINT | total_rqst_mapped_in - Total request mapped-in monitor element |
| TOTAL_RQST_MAPPED_OUT | BIGINT | total_rqst_mapped_out - Total request mapped-out monitor element |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time monitor element |
| TOTAL_WAIT_TIME | BIGINT | total_wait_time - Total wait time monitor element |
| APP_RQSTS_COMPLETED_TOTAL | BIGINT | app_rqsts_completed_total - Total application requests completed monitor element |
| TOTAL_SECTION_SORT_TIME | BIGINT | total_section_sort_time - Total section sort time monitor element |
| TOTAL_SECTION_SORT_PROC_TIME | BIGINT | total_section_sort_proc_time - Total section sort processing time monitor element |
| TOTAL_SECTION_SORTS | BIGINT | total_section_sorts - Total section sorts monitor element |
| TOTAL_SORTS | BIGINT | total_sorts - Total sorts monitor element |
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post threshold sorts monitor element |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post shared threshold sorts monitor element |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows monitor element |
| TOTAL_COMPILE_TIME | BIGINT | total_compile_time - Total compile time monitor element |
| TOTAL_COMPILE_PROC_TIME | BIGINT | total_compile_proc_time - Total compile processing time monitor element |
| TOTAL_COMPILATIONS | BIGINT | total_compilations - Total compilations monitor element |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|----------------------------------|-----------|---|
| TOTAL_IMPLICIT_COMPILE_TIME | BIGINT | total_implicit_compile_time - Total implicit compile time monitor element |
| TOTAL_IMPLICIT_COMPILE_PROC_TIME | BIGINT | total_implicit_compile_proc_time - Total implicit compile processing time monitor element |
| TOTAL_IMPLICIT_COMPILEMENTS | BIGINT | total_implicit_compilations - Total implicit complications monitor element |
| TOTAL_SECTION_TIME | BIGINT | total_section_time - Total section time monitor element |
| TOTAL_SECTION_PROC_TIME | BIGINT | total_section_proc_time - Total section processing time monitor element |
| TOTAL_APP_SECTION_EXECUTIONS | BIGINT | total_app_section_executions - Total application section executions monitor element |
| TOTAL_ACT_TIME | BIGINT | total_act_time - Total activity time monitor element |
| TOTAL_ACT_WAIT_TIME | BIGINT | total_act_wait_time - Total activity wait time monitor element |
| ACT_RQSTS_TOTAL | BIGINT | act_rqsts_total - Total activity requests monitor elements |
| TOTAL_ROUTINE_TIME | BIGINT | total_routine_time - Total routine time monitor element |
| TOTAL_ROUTINE_INVOCATIONS | BIGINT | total_routine_invocations - Total routine invocations monitor elements |
| TOTAL_COMMIT_TIME | BIGINT | total_commit_time - Total commit time monitor element |
| TOTAL_COMMIT_PROC_TIME | BIGINT | total_commit_proc_time - Total commits processing time monitor element |
| TOTAL_APP_COMMITS | BIGINT | total_app_commits - Total application commits monitor elements |
| INT_COMMITS | BIGINT | int_commits - Internal commits monitor element |
| TOTAL_ROLLBACK_TIME | BIGINT | total_rollback_time - Total rollback time monitor element |
| TOTAL_ROLLBACK_PROC_TIME | BIGINT | total_rollback_proc_time - Total rollback processing time monitor element |
| TOTAL_APP_ROLLBACKS | BIGINT | total_app_rollbacks - Total application rollbacks monitor element |
| INT_ROLLBACKS | BIGINT | int_rollbacks - Internal rollbacks monitor element |
| TOTAL_RUNSTATS_TIME | BIGINT | total_runstats_time - Total runtime statistics time monitor element |
| TOTAL_RUNSTATS_PROC_TIME | BIGINT | total_runstats_proc_time - Total runtime statistics processing time monitor element |
| TOTAL_RUNSTATS | BIGINT | total_runstats - Total runtime statistics monitor element |
| TOTAL_REORG_TIME | BIGINT | total_reorg_time - Total reorganization time monitor element |
| TOTAL_REORG_PROC_TIME | BIGINT | total_reorg_proc_time - Total reorganization processing time monitor element |
| TOTAL_REORGS | BIGINT | total_reorgs - Total reorganizations monitor element |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|--------------------------------|-----------|--|
| TOTAL_LOAD_TIME | BIGINT | total_load_time - Total load time monitor element |
| TOTAL_LOAD_PROC_TIME | BIGINT | total_load_proc_time - Total load processing time monitor element |
| TOTAL_LOADS | BIGINT | total_loads - Total loads monitor element |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts monitor element |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups monitor element |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts monitor element |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups monitor element |
| THRESH_VIOLATIONS | BIGINT | thresh_violations - Number of threshold violations monitor element |
| NUM_LW_THRESH_EXCEEDED | BIGINT | num_lw_thresh_exceeded - Number of lock wait thresholds exceeded monitor element |
| LOCK_WAITS_GLOBAL | BIGINT | lock_waits_global - Lock waits global monitor element |
| LOCK_WAIT_TIME_GLOBAL | BIGINT | lock_wait_time_global - Lock wait time global monitor element |
| LOCK_TIMEOUTS_GLOBAL | BIGINT | lock_timeouts_global - Lock timeouts global monitor element |
| LOCK_ESCALS_MAXLOCKS | BIGINT | lock_escals_maxlocks - Number of maxlocks lock escalations monitor element |
| LOCK_ESCALS_LOCKLIST | BIGINT | lock_escals_locklist - Number of locklist lock escalations monitor element |
| LOCK_ESCALS_GLOBAL | BIGINT | lock_escals_global - Number of global lock escalations monitor element |
| RECLAIM_WAIT_TIME | BIGINT | reclaim_wait_time - Reclaim wait time monitor element |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME | BIGINT | spacemappage_reclaim_wait_time - Space map page reclaim wait time monitor element |
| CF_WAITS | BIGINT | cf_waits - Number of cluster caching facility waits monitor element |
| CF_WAIT_TIME | BIGINT | cf_wait_time - cluster caching facility wait time monitor element |
| POOL_DATA_GBP_L_READS | BIGINT | pool_data_gbp_l_reads - Group buffer pool data logical reads monitor element |
| POOL_DATA_GBP_P_READS | BIGINT | pool_data_gbp_p_reads - Group buffer pool data physical reads monitor element |
| POOL_DATA_LBP_PAGES_FOUND | BIGINT | pool_data_lbp_pages_found - Local buffer pool found data pages monitor element |
| POOL_DATA_GBP_INVALID_PAGES | BIGINT | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages monitor element |
| POOL_INDEX_GBP_L_READS | BIGINT | pool_index_gbp_l_reads - Group buffer pool index logical reads monitor element |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|------------------------------|-----------|--|
| POOL_INDEX_GBP_P_READS | BIGINT | pool_index_gbp_p_reads - Group buffer pool index physical reads monitor elements |
| POOL_INDEX_LBP_PAGES_FOUND | BIGINT | pool_index_lbp_pages_found - Local buffer pool index pages found monitor element |
| POOL_INDEX_GBP_INVALID_PAGES | BIGINT | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages monitor element |
| POOL_XDA_GBP_L_READS | BIGINT | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests monitor element |
| POOL_XDA_GBP_P_READS | BIGINT | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests monitor element |
| POOL_XDA_LBP_PAGES_FOUND | BIGINT | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found monitor element |
| POOL_XDA_GBP_INVALID_PAGES | BIGINT | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages monitor element |
| AUDIT_EVENTS_TOTAL | BIGINT | audit_events_total - Total audit events monitor element |
| AUDIT_FILE_WRITES_TOTAL | BIGINT | audit_file_writes_total - Total audit files written monitor element |
| AUDIT_FILE_WRITE_WAIT_TIME | BIGINT | audit_file_write_wait_time - Audit file write wait time monitor element |
| AUDIT_SUBSYSTEM_WAITS_TOTAL | BIGINT | audit_subsystem_waits_total - Total audit subsystem waits monitor element |
| AUDIT_SUBSYSTEM_WAIT_TIME | BIGINT | audit_subsystem_wait_time - Audit subsystem wait time monitor element |
| DIAGLOG_WRITES_TOTAL | BIGINT | diaglog_writes_total - Total diagnostic log file writes monitor element |
| DIAGLOG_WRITE_WAIT_TIME | BIGINT | diaglog_write_wait_time - Diagnostic log file write wait time monitor element |
| FCM_MESSAGE_RECVS_TOTAL | BIGINT | fcm_message_recvs_total - Total FCM message receives monitor element |
| FCM_MESSAGE_RECV_VOLUME | BIGINT | fcm_message_recv_volume - FCM message received volume monitor element |
| FCM_MESSAGE_RECV_WAIT_TIME | BIGINT | fcm_message_recv_wait_time - FCM message received wait time monitor element |
| FCM_MESSAGE_SENDS_TOTAL | BIGINT | fcm_message_sends_total - Total FCM message sends monitor element |
| FCM_MESSAGE_SEND_VOLUME | BIGINT | fcm_message_send_volume - FCM message send volume monitor element |
| FCM_MESSAGE_SEND_WAIT_TIME | BIGINT | fcm_message_send_wait_time - FCM message send wait time monitor element |
| FCM_TQ_RECVS_TOTAL | BIGINT | fcm_tq_recvs_total - FCM table queue receives total monitor element |
| FCM_TQ_RECV_VOLUME | BIGINT | fcm_tq_recv_volume - FCM table queue received volume monitor element |
| FCM_TQ_RECV_WAIT_TIME | BIGINT | fcm_tq_recv_wait_time - FCM table queue received wait time monitor element |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|-----------------------------------|-----------|--|
| FCM_TQ_SENDS_TOTAL | BIGINT | fcm_tq_sends_total - FCM table queue send total monitor element |
| FCM_TQ_SEND_VOLUME | BIGINT | fcm_tq_send_volume - FCM table queue send volume monitor element |
| FCM_TQ_SEND_WAIT_TIME | BIGINT | fcm_tq_send_wait_time - FCM table queue send wait time monitor element |
| TOTAL_ROUTINE_USER_CODE_PROC_TIME | BIGINT | total_routine_user_code_proc_time - Total routine user code processing time monitor element |
| TOTAL_ROUTINE_USER_CODE_TIME | BIGINT | total_routine_user_code_time - Total routine user code time monitor element |
| TQ_TOT_SEND_SPILLS | BIGINT | tq_tot_send_spills - Total number of table queue buffers overflowed monitor element |
| EVMON_WAIT_TIME | BIGINT | evmon_wait_time - Event monitor wait time monitor element |
| EVMON_WAITS_TOTAL | BIGINT | evmon_waits_total - Event monitor total waits monitor element |
| TOTAL_EXTENDED_LATCH_WAIT_TIME | BIGINT | total_extended_latch_wait_time - Total extended latch wait time monitor element |
| TOTAL_EXTENDED_LATCH_WAITS | BIGINT | total_extended_latch_waits - Total extended latch waits monitor element |
| TOTAL_STATS_FABRICATION_TIME | BIGINT | total_stats_fabrication_time - Total statistics fabrication time monitor element |
| TOTAL_STATS_FABRICATION_PROC_TIME | BIGINT | total_stats_fabrication_proc_time - Total statistics fabrication processing time monitor element |
| TOTAL_STATS_FABRICATIONS | BIGINT | total_stats_fabrications - Total statistics fabrications monitor elements |
| TOTAL_SYNC_RUNSTATS_TIME | BIGINT | total_sync_runstats_time - Total synchronous RUNSTATS time monitor elements |
| TOTAL_SYNC_RUNSTATS_PROC_TIME | BIGINT | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time monitor element |
| TOTAL_SYNC_RUNSTATS | BIGINT | total_sync_runstats - Total synchronous RUNSTATS activities monitor element |
| TOTAL_DISP_RUN_QUEUE_TIME | BIGINT | total_disp_run_queue_time - Total dispatcher run queue time monitor element |
| TOTAL_PEDS | BIGINT | total_peds - Total partial early distincts monitor element |
| DISABLED_PEDS | BIGINT | disabled_peds - Disabled partial early distincts monitor element |
| POST_THRESHOLD_PEDS | BIGINT | post_threshold_peds - Partial early distincts threshold monitor element |
| TOTAL_PEAS | BIGINT | total_peas - Total partial early aggregations monitor element |
| POST_THRESHOLD_PEAS | BIGINT | post_threshold_peas - Partial early aggregation threshold monitor element |
| TQ_SORT_HEAP_REQUESTS | BIGINT | tq_sort_heap_requests - Table queue sort heap requests monitor element |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|------------------------------------|-----------|--|
| TQ_SORT_HEAP_REJECTIONS | BIGINT | tq_sort_heap_rejections - Table queue sort heap rejections monitor element |
| POOL_QUEUED_ASYNC_DATA_REQS | BIGINT | pool_queued_async_data_reqs - Data prefetch requests monitor element |
| POOL_QUEUED_ASYNC_INDEX_REQS | BIGINT | pool_queued_async_index_reqs - Index prefetch requests monitor element |
| POOL_QUEUED_ASYNC_XDA_REQS | BIGINT | pool_queued_async_xda_reqs - XDA prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_OTHER_REQS | BIGINT | pool_queued_async_other_reqs - Non-prefetch requests monitor element |
| POOL_QUEUED_ASYNC_DATA_PAGES | BIGINT | pool_queued_async_data_pages - Data pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_INDEX_PAGES | BIGINT | pool_queued_async_index_pages - Index pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_XDA_PAGES | BIGINT | pool_queued_async_xda_pages - XDA pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES | BIGINT | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES | BIGINT | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_DATA_REQS | BIGINT | pool_failed_async_data_reqs - Failed data prefetch requests monitor element |
| POOL_FAILED_ASYNC_INDEX_REQS | BIGINT | pool_failed_async_index_reqs - Failed index prefetch requests monitor element |
| POOL_FAILED_ASYNC_XDA_REQS | BIGINT | pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element |
| POOL_FAILED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|---|-----------|--|
| POOL_FAILED_ASYNC_OTHER_REQS | BIGINT | pool_failed_async_other_reqs - Failed non-prefetch requests monitor element |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch monitor element |
| PREFETCH_WAITS | BIGINT | prefetch_waits - Prefetcher wait count monitor element |
| APP_ACT_COMPLETED_TOTAL | BIGINT | app_act_completed_total - Total successful external coordinator activities monitor element |
| APP_ACT_ABORTED_TOTAL | BIGINT | app_act_aborted_total - Total failed external coordinator activities monitor element |
| APP_ACT_REJECTED_TOTAL | BIGINT | app_act_rejected_total - Total rejected external coordinator activities monitor element |
| TOTAL_CONNECT_REQUEST_TIME | BIGINT | total_connect_request_time - Total connection or switch user request time monitor element |
| TOTAL_CONNECT_REQUEST_PROC_TIME | BIGINT | total_connect_request_proc_time - Total connection or switch user request processing time monitor element |
| TOTAL_CONNECT_REQUESTS | BIGINT | total_connect_requests - Connection or switch user requests monitor element |
| TOTAL_CONNECT_AUTHENTICATION_TIME | BIGINT | total_connect_authentication_time - Total connection or switch user authentication request time monitor element |
| TOTAL_CONNECT_AUTHENTICATION_PROC_TIME | BIGINT | total_connect_authentication_proc_time - Total connection authentication processing time monitor element |
| TOTAL_CONNECT_AUTHENTIFICATIONS | BIGINT | total_connect_authentications - Connections or switch user authentications performed monitor element |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |
| COMM_EXIT_WAIT_TIME | BIGINT | comm_exit_wait_time - Communication buffer exit wait time monitor element |
| COMM_EXIT_WAITS | BIGINT | comm_exit_waits - Communication buffer exit number of waits monitor element |
| IDA_SEND_WAIT_TIME | BIGINT | ida_send_wait_time - Time spent waiting to send data monitor element |
| IDA_SENDS_TOTAL | BIGINT | ida_sends_total - Number of times data sent monitor element |
| IDA_SEND_VOLUME | BIGINT | ida_send_volume - Total data volume sent monitor element |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|------------------------------|-----------|---|
| IDA_RECV_WAIT_TIME | BIGINT | ida_recv_wait_time - Time spent waiting to receive data monitor element |
| IDA_RECVS_TOTAL | BIGINT | ida_recv_total - Number of times data received monitor element |
| IDA_RECV_VOLUME | BIGINT | ida_recv_volume - Total data volume received monitor element |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted monitor element |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted monitor element |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated monitor element |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total Hash Joins monitor element |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total Hash Loops monitor element |
| HASH_JOIN_OVERFLOW | BIGINT | hash_join_overflows - Hash Join Overflows monitor element |
| HASH_JOIN_SMALL_OVERFLOW | BIGINT | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| POST_SHRTHRESHOLD_HASH_JOINS | BIGINT | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP Functions monitor element |
| OLAP_FUNC_OVERFLOW | BIGINT | olap_func_overflows - OLAP Function Overflows monitor element |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element |
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL Statements Attempted monitor element |
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed Statement Operations monitor element |
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL Statements Executed monitor element |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| MERGE_SQL_STMTS | BIGINT | merge_sql_stmts - Merge SQL statements executed |
| XQUERY_STMTS | BIGINT | xquery_stmts - XQuery Statements Attempted monitor element |
| IMPLICIT_REBINDS | BIGINT | implicit_rebinds - number of implicit rebinds |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/Precompiles Attempted monitor element |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal Rows Inserted monitor element |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal Rows Updated monitor element |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|---------------------------------------|-----------|---|
| CALL_SQL_STMTS | BIGINT | call_sql_stmts - CALL SQL statements executed |
| POOL_COL_L_READS | BIGINT | pool_col_l_reads - Buffer pool column-organized logical reads |
| POOL_TEMP_COL_L_READS | BIGINT | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| POOL_COL_P_READS | BIGINT | pool_col_p_reads - Buffer pool column-organized physical reads |
| POOL_TEMP_COL_P_READS | BIGINT | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| POOL_COL_LBP_PAGES_FOUND | BIGINT | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found |
| POOL_COL_WRITES | BIGINT | pool_col_writes - Buffer pool column-organized writes |
| POOL_COL_GBP_L_READS | BIGINT | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| POOL_COL_GBP_P_READS | BIGINT | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| POOL_COL_GBP_INVALID_PAGES | BIGINT | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |
| POOL_QUEUED_ASYNC_COL_REQS | BIGINT | pool_queued_async_col_reqs - Column-organized prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_REQS | BIGINT | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| POOL_QUEUED_ASYNC_COL_PAGES | BIGINT | pool_queued_async_col_pages - Column-organized page prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_PAGES | BIGINT | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| POOL_FAILED_ASYNC_COL_REQS | BIGINT | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| POOL_FAILED_ASYNC_TEMP_COL_REQS | BIGINT | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |
| TOTAL_COL_TIME | BIGINT | total_col_time - Total column-organized time |
| TOTAL_COL_PROC_TIME | BIGINT | total_col_proc_time - Total column-organized processing time |
| TOTAL_COL_EXECUTIONS | BIGINT | total_col_executions - Total column-organized executions |
| POST_THRESHOLD_HASH_JOINS | BIGINT | post_threshold_hash_joins - Hash Join Threshold monitor element |
| POOL_DATA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|---|-----------|--------------------------|
| POOL_XDA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_READ_TIME | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_WRITE_TIME | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|---|-----------|--|
| POOL_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| TOTAL_HASH_GRPBY | BIGINT | total_hash_grpbys - Total hash group by operations |
| HASH_GRPBY_OVERFLOW | BIGINT | hash_grpby_overflows - Hash group by overflows |
| POST_THRESHOLD_HASH_GRPBY | BIGINT | post_threshold_hash_grpbys - Hash group by threshold |
| POST_THRESHOLD_OLAP_FUNCS | BIGINT | post_threshold_olap_funcs - OLAP function threshold monitor element |
| POST_THRESHOLD_COL_VECTOR_CONSUMERS | BIGINT | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| TOTAL_COL_VECTOR_CONSUMERS | BIGINT | total_col_vector_consumers - Total columnar_vector consumers |
| ACTIVE_HASH_GRPBY | BIGINT | active_hash_grpbys - Active hash GROUP BY operations |
| ACTIVE_HASH_JOINS | BIGINT | active_hash_joins - Active hash joins |
| ACTIVE_OLAP_FUNCS | BIGINT | active_olap_funcs - Active OLAP Functions |
| ACTIVE_PEAS | BIGINT | active_peas - Active partial early aggregations operations |
| ACTIVE_PEDS | BIGINT | active_peds - Active partial early distinct operations |
| ACTIVE_SORT_CONSUMERS | BIGINT | active_sort_consumers - Active sort consumers |
| ACTIVE_SORTS | BIGINT | active_sorts - Active Sorts |
| ACTIVE_COL_VECTOR_CONSUMERS | BIGINT | active_col_vector_consumers - Active columnar_vector consumers |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total Sort Heap Allocated |
| SORT_SHRHEAP_ALLOCATED | BIGINT | sort_shrheap_allocated - Sort Share Heap Currently Allocated |
| TOTAL_BACKUP_TIME | BIGINT | total_backup_time - Total elapsed time for online backups |
| TOTAL_BACKUP_PROC_TIME | BIGINT | total_backup_proc_time - Total non-wait time for online backups |
| TOTAL_BACKUPS | BIGINT | total_backups - Total backups |
| TOTAL_INDEX_BUILD_TIME | BIGINT | total_index_build_time - Total time spent building indexes due to index creation, recreation or rebuilding |
| TOTAL_INDEX_BUILD_PROC_TIME | BIGINT | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation, recreation or rebuilding |
| TOTAL_INDEXES_BUILT | BIGINT | total_indexes_built - Total number of indexes built |
| FCM_TQ_RECV_WAITS_TOTAL | BIGINT | fcm_tq_recv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| FCM_MESSAGE_RECV_WAITS_TOTAL | BIGINT | fcm_message_recv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| FCM_TQ_SEND_WAITS_TOTAL | BIGINT | fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |

Table 157. Information returned for MON_GET_SERVICE_SUBCLASS (continued)

| Column Name | Data Type | Description |
|--------------------------------|-----------|--|
| FCM_MESSAGE_SEND_WAITS_TOTAL | BIGINT | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| FCM_SEND_WAITS_TOTAL | BIGINT | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| FCM_RECV_WAITS_TOTAL | BIGINT | fcm_recv_waits_total - Number of times spent waiting to receive data through FCM monitor element |
| EXT_TABLE_RECV_WAIT_TIME | BIGINT | ext_table_recv_wait_time - Total agent wait time for external table readers monitor element |
| EXT_TABLE_RECVS_TOTAL | BIGINT | ext_table_recvs_total - Total row batches received from external table readers monitor element |
| EXT_TABLE_RECV_VOLUME | BIGINT | ext_table_recv_volume - Total data received from external table readers monitor element |
| EXT_TABLE_READ_VOLUME | BIGINT | ext_table_read_volume - Total data read by external table readers monitor element |
| EXT_TABLE_SEND_WAIT_TIME | BIGINT | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| EXT_TABLE_SENDS_TOTAL | BIGINT | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| EXT_TABLE_SEND_VOLUME | BIGINT | ext_table_send_volume - Total data sent to external table writers monitor element |
| EXT_TABLE_WRITE_VOLUME | BIGINT | ext_table_write_volume - Total data written by external table writers monitor element |
| COL_VECTOR_CONSUMER_OVERFLOWES | BIGINT | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |
| TOTAL_COL_SYNOPSIS_TIME | BIGINT | total_col_synopsis_time - Total column-organized synopsis table time monitor element |
| TOTAL_COL_SYNOPSIS_PROC_TIME | BIGINT | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| TOTAL_COL_SYNOPSIS_EXECUTIONS | BIGINT | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |
| COL_SYNOPSIS_ROWS_INSERTED | BIGINT | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| LOB_PREFETCH_WAIT_TIME | BIGINT | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| LOB_PREFETCH_REQS | BIGINT | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| FED_ROWS_DELETED | BIGINT | fed_rows_deleted - Rows deleted by a federation system monitor element |
| FED_ROWS_INSERTED | BIGINT | fed_rows_inserted - Rows inserted by a federation system monitor element |
| FED_ROWS_UPDATED | BIGINT | fed_rows_updated - Rows updated by a federation system monitor element |

Table 157. Information returned for `MON_GET_SERVICE_SUBCLASS` (continued)

| Column Name | Data Type | Description |
|----------------------|-----------|---|
| FED_ROWS_READ | BIGINT | fed_rows_read - Rows read by a federation system monitor element |
| FED_WAIT_TIME | BIGINT | fed_wait_time - Time spent by a federation server monitor element |
| FED_WAITS_TOTAL | BIGINT | fed_waits_total - Total number of execution times for a federation server monitor element |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Insert count for SQL workspace monitor element |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section lookups - SQL workspace access count monitor element |
| ADM_OVERFLOWES | BIGINT | Reserved for future use. |
| ADM_BYPASS_ACT_TOTAL | BIGINT | Reserved for future use. |

Usage notes

The metrics returned by the `MON_GET_SERVICE_SUBCLASS` table function represent the accumulation of all metrics for requests that have executed under the indicated service subclass. Metrics are rolled up to a service class on unit of work boundaries, and periodically during the execution of requests. Therefore, the values reported by this table function reflect the current state of the system at the time of the most recent rollup. Metrics are strictly increasing in value. To determine the value of a given metric for an interval of time, use the `MON_GET_SERVICE_SUBCLASS` table function to query the metric at the start and end of the interval, and compute the difference.

Request metrics are controlled through the `COLLECT REQUEST METRICS` clause on service superclasses and the `mon_req_metrics` database configuration parameter at the database level. Metrics are only collected for a request if the request is processed by an agent in a service subclass whose parent service superclass has request metrics enabled, or if request metrics collection is enabled for the entire database. By default, request metrics are enabled at the database level. If request metrics are disabled at the database level and for a service superclass, the metrics reported for each connection mapped to that service superclass stop increasing (or remain at 0 if request metrics were disabled at database activation time).

The `MON_GET_SERVICE_SUBCLASS` table function returns one row of data per service subclass and per member. No aggregation across service classes (on a member), or across members (for a service class or more), is performed. However, aggregation can be achieved through SQL queries as shown in the example. The input parameters have the effect of being ANDed together. Therefore, if you specify conflicting input parameters (for example, a superclass name SUPA and subclass name SUBB that is not a subclass of SUPA), no rows are returned.

Tip: A request might execute in more than one service subclass. For example, this situation might occur if a request is mapped from one service subclass to another by using a Workload Manager (WLM) threshold with a `REMAP ACTIVITY` action. Although the time spent metrics are updated for each service subclass under which the request executes, the request counters are incremented for the service subclass where the request completed. Therefore, you should not analyze the averages of request times for a single subclass. All subclasses to which an activity can be mapped must be analyzed in conjunction with one another. For example, if a

threshold exists that can map activities from service subclass A to service subclass B, then when you compute averages of requests, you should aggregate the counters and metrics for service subclasses A and B, and compute the averages using the aggregates.

Note that the metrics from background EDUs such as prefetchers and page cleaners are not rolled up to service classes.

Example

Display the total CPU time used and total number of requests processed for each service class, ordered by CPU usage.

```
SELECT varchar(service_superclass_name,30) as service_superclass,
        varchar(service_subclass_name,30) as service_subclass,
        sum(total_cpu_time) as total_cpu,
        sum(app_rqsts_completed_total) as total_rqsts
FROM TABLE(MON_GET_SERVICE_SUBCLASS('','',-2)) AS t
GROUP BY service_superclass_name, service_subclass_name
ORDER BY total_cpu desc
```

The following is an example of output from this query.

| SERVICE_SUPERCLASS | SERVICE_SUBCLASS | ... |
|----------------------------|--------------------|-----|
| ----- | ----- | ... |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | ... |
| SYSDEFAULTMAINTENANCECLASS | SYSDEFAULTSUBCLASS | ... |
| SYSDEFAULTSYSTEMCLASS | SYSDEFAULTSUBCLASS | ... |

3 record(s) selected.

Output for query (continued).

| ... TOTAL_CPU | TOTAL_RQSTS |
|---------------|-------------|
| ... | ----- |
| ... | 967673 100 |
| .. . | 0 0 |
| ... | 0 0 |

MON_GET_SERVICE_SUBCLASS_DETAILS table function - Get service subclass metrics as an XML document

The MON_GET_SERVICE_SUBCLASS_DETAILS table function returns metrics for one or more service subclasses.

The metrics are returned in an XML document.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

```
►►—MON_GET_SERVICE_SUBCLASS_DETAILS—(—service_superclass_name—,—————►  
►—service_subclass_name—,—member—)—————►◄
```

The schema is SYSPROC.

Routine parameters

service_superclass_name

An input argument of type VARCHAR(128) that specifies a valid service superclass name in the currently connected database when calling this function. If the argument is null or an empty string, metrics are retrieved for all the superclasses in the database.

service_subclass_name

An input argument of type VARCHAR(128) that specifies a valid service subclass name in the currently connected database when calling this function. If the argument is null or an empty string, metrics are retrieved for all the subclasses in the database.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

Information returned

Table 158. Information returned for MON_GET_SERVICE_SUBCLASS_DETAILS

| Column Name | Data Type | Description |
|-------------------------|--------------|--|
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name |
| SERVICE_SUBCLASS_NAME | VARCHAR(128) | service_subclass_name - Service subclass name |
| SERVICE_CLASS_ID | INTEGER | service_class_id - Service class ID |
| MEMBER | SMALLINT | member - Database member |
| DETAILS | BLOB(1M) | XML document that contains detailed metrics for the service class. See Table 159 on page 781 for a description of the elements in this document. |

The following example shows the structure of the XML document that is returned in the DETAILS column.

```
<db2_service_subclass xmlns="http://www.ibm.com/xmlns/prod/db2/mon" release="90700000">  
  <service_superclass_name>SYSDEFAULTSYSTEMCLASS</service_superclass_name>  
  <service_subclass_name>SYSDEFAULTSUBCLASS</service_subclass_name>  
  <service_subclass_id>11</service_subclass_id>  
  <member>0</member>  
  <system_metrics release="90700000">  
    <act_aborted_total>5</act_aborted_total>  
    ...  
    <wlm_queue_assignments_total>3</wlm_queue_assignments_total>  
  </system_metrics>  
</db2_service_subclass>
```

For the full schema, see `sql1lib/misc/DB2MonRoutines.xsd`.

Table 159. Detailed metrics returned for MON_GET_SERVICE_SUBCLASS_DETAILS

| Element Name | Data Type | Description or corresponding monitor element |
|-------------------------------|-----------------------|--|
| act_aborted_total | xs:nonNegativeInteger | act_aborted_total - Total aborted activities |
| act_completed_total | xs:nonNegativeInteger | act_completed_total - Total completed activities |
| act_rejected_total | xs:nonNegativeInteger | act_rejected_total - Total rejected activities |
| act_rqsts_total | xs:nonNegativeInteger | act_rqsts_total - Total activity requests |
| active_col_vector_consumers | xs:long | active_col_vector_consumers - Active columnar_vector consumers |
| active_hash_grpbys | xs:long | active_hash_grpbys - Active hash GROUP BY operations |
| active_hash_joins | xs:long | active_hash_joins - Active hash joins |
| active_olap_funcs | xs:long | active_olap_funcs - Active OLAP Functions |
| active_peas | xs:long | active_peas - Active partial early aggregations operations |
| active_peds | xs:long | active_peds - Active partial early distinct operations |
| active_sort_consumers | xs:long | active_sort_consumers - Active sort consumers |
| active_sorts | xs:long | active_sorts - Active Sorts |
| adm_bypass_act_total | xs:long | Reserved for future use. |
| adm_overflows | xs:long | Reserved for future use. |
| agent_wait_time | xs:nonNegativeInteger | agent_wait_time - Agent wait time |
| agent_waits_total | xs:nonNegativeInteger | agent_waits_total - Total agent waits |
| app_act_aborted_total | xs:nonNegativeInteger | app_act_aborted_total - Total failed external coordinator activities monitor element |
| app_act_completed_total | xs:nonNegativeInteger | app_act_completed_total - Total successful external coordinator activities monitor element |
| app_act_rejected_total | xs:nonNegativeInteger | app_act_rejected_total - Total rejected external coordinator activities monitor element |
| app_rqsts_completed_total | xs:nonNegativeInteger | app_rqsts_completed_total - Total application requests completed |
| appl_section_inserts | xs:nonNegativeInteger | appl_section_inserts - Insert count for SQL workspace monitor element |
| appl_section_lookups | xs:nonNegativeInteger | appl_section_lookups - SQL workspace access count monitor element |
| audit_events_total | xs:nonNegativeInteger | audit_events_total - Total audit events |
| audit_file_write_wait_time | xs:nonNegativeInteger | audit_file_write_wait_time - Audit file write wait time |
| audit_file_writes_total | xs:nonNegativeInteger | audit_file_writes_total - Total Audit files written |
| audit_subsystem_wait_time | xs:nonNegativeInteger | audit_subsystem_wait_time - Audit subsystem wait time |
| audit_subsystem_waits_total | xs:nonNegativeInteger | audit_subsystem_waits_total - Total audit subsystem waits |
| binds_precompiles | xs:long | binds_precompiles - Binds/Precompiles Attempted monitor element |
| call_sql_stmts | xs:long | call_sql_stmts - CALL SQL statements executed monitor element |
| cat_cache_inserts | xs:nonNegativeInteger | cat_cache_inserts - Catalog cache inserts |
| cat_cache_lookups | xs:nonNegativeInteger | cat_cache_lookups - Catalog cache lookups |
| cf_wait_time | xs:long | cf_wait_time - cluster caching facility wait time monitor element |
| cf_waits | xs:long | cf_waits - Number of cluster caching facility waits monitor element |
| client_idle_wait_time | xs:nonNegativeInteger | client_idle_wait_time - Client idle wait time |
| col_synopsis_rows_inserted | xs:nonNegativeInteger | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| col_vector_consumer_overflows | xs:long | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |
| comm_exit_wait_time | xs:nonNegativeInteger | comm_exit_wait_time - Communication buffer exit wait time monitor element |

Table 159. Detailed metrics returned for MON_GET_SERVICE_SUBCLASS_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|------------------------------|-----------------------|--|
| comm_exit_waits | xs:nonNegativeInteger | comm_exit_waits - Communication buffer exit number of waits monitor element |
| ddl_sql_stmts | xs:long | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| deadlocks | xs:nonNegativeInteger | deadlocks - Deadlocks detected |
| diaglog_write_wait_time | xs:nonNegativeInteger | diaglog_write_wait_time - Diag log write time |
| diaglog_writes_total | xs:nonNegativeInteger | diaglog_writes_total - Diag log total writes |
| direct_read_reqs | xs:nonNegativeInteger | direct_read_reqs - Direct read requests |
| direct_read_time | xs:nonNegativeInteger | direct_read_time - Direct read time |
| direct_reads | xs:nonNegativeInteger | direct_reads - Direct reads from database |
| direct_write_reqs | xs:nonNegativeInteger | direct_write_reqs - Direct write requests |
| direct_write_time | xs:nonNegativeInteger | direct_write_time - Direct write time |
| direct_writes | xs:nonNegativeInteger | direct_writes - Direct writes to database |
| disabled_peds | xs:long | disabled_peds - Disabled partial early distincts |
| dynamic_sql_stmts | xs:long | dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element |
| evmon_wait_time | xs:nonNegativeInteger | evmon_wait_time - Event monitor wait time |
| evmon_waits_total | xs:nonNegativeInteger | evmon_waits_total - Event monitor total waits |
| ext_table_read_volume | xs:nonNegativeInteger | ext_table_read_volume - Total data read by external table readers monitor element |
| ext_table_rcv_volume | xs:nonNegativeInteger | ext_table_rcv_volume - Total data received from external table readers monitor element |
| ext_table_rcv_wait_time | xs:nonNegativeInteger | ext_table_rcv_wait_time - Total agent wait time for external table readers monitor element |
| ext_table_rcvs_total | xs:nonNegativeInteger | ext_table_rcvs_total - Total row batches received from external table readers monitor element |
| ext_table_send_volume | xs:nonNegativeInteger | ext_table_send_volume - Total data sent to external table writers monitor element |
| ext_table_send_wait_time | xs:nonNegativeInteger | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| ext_table_sends_total | xs:nonNegativeInteger | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| ext_table_write_volume | xs:nonNegativeInteger | ext_table_write_volume - Total data written by external table writers monitor element |
| failed_sql_stmts | xs:long | failed_sql_stmts - Failed Statement Operations monitor element |
| fcm_message_rcv_volume | xs:nonNegativeInteger | fcm_message_rcv_volume - FCM message rcv volume |
| fcm_message_rcv_wait_time | xs:nonNegativeInteger | fcm_message_rcv_wait_time - FCM message rcv wait time |
| fcm_message_rcv_waits_total | xs:nonNegativeInteger | fcm_message_rcv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| fcm_message_rcvs_total | xs:nonNegativeInteger | fcm_message_rcvs_total - FCM message rcvs total |
| fcm_message_send_volume | xs:nonNegativeInteger | fcm_message_send_volume - FCM message send volume |
| fcm_message_send_wait_time | xs:nonNegativeInteger | fcm_message_send_wait_time - FCM message send wait time |
| fcm_message_send_waits_total | xs:nonNegativeInteger | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| fcm_message_sends_total | xs:nonNegativeInteger | fcm_message_sends_total - FCM message sends total |
| fcm_rcv_volume | xs:nonNegativeInteger | fcm_rcv_volume - FCM rcv volume |
| fcm_rcv_wait_time | xs:nonNegativeInteger | fcm_rcv_wait_time - FCM rcv wait time |
| fcm_rcv_waits_total | xs:nonNegativeInteger | fcm_rcv_waits_total - Number of times spent waiting to receive data through FCM monitor element |

Table 159. Detailed metrics returned for MON_GET_SERVICE_SUBCLASS_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|---------------------------|-----------------------|--|
| fcm_recvs_total | xs:nonNegativeInteger | fcm_recvs_total - FCM recvs total |
| fcm_send_volume | xs:nonNegativeInteger | fcm_send_volume - FCM send volume |
| fcm_send_wait_time | xs:nonNegativeInteger | fcm_send_wait_time - FCM send wait time |
| fcm_send_waits_total | xs:nonNegativeInteger | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| fcm_sends_total | xs:nonNegativeInteger | fcm_sends_total - FCM sends total |
| fcm_tq_recv_volume | xs:nonNegativeInteger | fcm_tq_recv_volume - FCM tablequeue recv volume |
| fcm_tq_recv_wait_time | xs:nonNegativeInteger | fcm_tq_recv_wait_time - FCM tablequeue recv wait time |
| fcm_tq_recv_waits_total | xs:nonNegativeInteger | fcm_tq_recv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| fcm_tq_recvs_total | xs:nonNegativeInteger | fcm_tq_recvs_total - FCM tablequeue recvs total |
| fcm_tq_send_volume | xs:nonNegativeInteger | fcm_tq_send_volume - FCM tablequeue send volume |
| fcm_tq_send_wait_time | xs:nonNegativeInteger | fcm_tq_send_wait_time - FCM tablequeue send wait time |
| fcm_tq_sends_total | xs:nonNegativeInteger | fcm_tq_sends_total - FCM tablequeue send total |
| fed_rows_deleted | xs:nonNegativeInteger | fed_rows_deleted - Rows deleted by a federation system monitor element |
| fed_rows_inserted | xs:nonNegativeInteger | fed_rows_inserted - Rows inserted by a federation system monitor element |
| fed_rows_read | xs:nonNegativeInteger | fed_rows_read - Rows read by a federation system monitor element |
| fed_rows_updated | xs:nonNegativeInteger | fed_rows_updated - Rows updated by a federation system monitor element |
| fed_wait_time | xs:nonNegativeInteger | fed_wait_time - Time spent by a federation server monitor element |
| fed_waits_total | xs:nonNegativeInteger | fed_waits_total - Total number of execution times for a federation server monitor element |
| hash_grpby_overflows | xs:long | hash_grpby_overflows - Hash group by overflows |
| hash_join_overflows | xs:long | hash_join_overflows - Hash Join Overflows monitor element |
| hash_join_small_overflows | xs:long | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| ida_recv_volume | xs:nonNegativeInteger | ida_recv_volume - Total data volume received monitor element

The total volume of data the database server received from an in-database analytics process. The value is reported in bytes. |
| ida_recv_wait_time | xs:nonNegativeInteger | ida_recv_wait_time - Time spent waiting to receive data monitor element

The total amount of time spent waiting to receive data from an in-database analytics process. |
| ida_recvs_total | xs:nonNegativeInteger | ida_recvs_total - Number of times data received monitor element

The total number of times data was received from an in-database analytics process. |
| ida_send_volume | xs:nonNegativeInteger | ida_send_volume - Total data volume sent monitor element

The total volume of data sent from the database server to an in-database analytics process. The value is reported in bytes. |
| ida_send_wait_time | xs:nonNegativeInteger | ida_send_wait_time - Time spent waiting to send data monitor element

The total amount of time spent waiting to send data to an in-database analytics process. |

Table 159. Detailed metrics returned for MON_GET_SERVICE_SUBCLASS_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|---------------------------------------|-----------------------|---|
| ida_sends_total | xs:nonNegativeInteger | ida_sends_total - Number of times data sent monitor element

The total number of times data was sent to an in-database analytics process. |
| implicit_rebinds | xs:long | implicit_rebinds - Number of implicit rebinds monitor element |
| int_commits | xs:nonNegativeInteger | int_commits - Internal commits |
| int_rollbacks | xs:nonNegativeInteger | int_rollbacks - Internal rollbacks |
| int_rows_deleted | xs:long | int_rows_deleted - Internal Rows Deleted monitor element |
| int_rows_inserted | xs:long | int_rows_inserted - Internal Rows Inserted monitor element |
| int_rows_updated | xs:long | int_rows_updated - Internal Rows Updated monitor element |
| ipc_rcv_volume | xs:nonNegativeInteger | ipc_rcv_volume - Interprocess communication rcv volume |
| ipc_rcv_wait_time | xs:nonNegativeInteger | ipc_rcv_wait_time - Interprocess communication rcv wait time |
| ipc_rcvs_total | xs:nonNegativeInteger | ipc_rcvs_total - Interprocess communication rcvs total |
| ipc_send_volume | xs:nonNegativeInteger | ipc_send_volume - Interprocess communication send volume |
| ipc_send_wait_time | xs:nonNegativeInteger | ipc_send_wait_time - Interprocess communication send wait time |
| ipc_sends_total | xs:nonNegativeInteger | ipc_sends_total - Interprocess communication send total |
| lob_prefetch_reqs | xs:nonNegativeInteger | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| lob_prefetch_wait_time | xs:nonNegativeInteger | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| lock_escals | xs:nonNegativeInteger | lock_escals - Number of lock escalations |
| lock_escals_global | xs:long | lock_escals_global - Number of global lock escalations monitor element |
| lock_escals_locklist | xs:long | lock_escals_locklist - Number of locklist lock escalations monitor element |
| lock_escals_maxlocks | xs:long | lock_escals_maxlocks - Number of maxlocks lock escalations monitor element |
| lock_timeouts | xs:nonNegativeInteger | lock_timeouts - Number of lock timeouts |
| lock_timeouts_global | xs:long | lock_timeouts_global - Lock timeouts global monitor element |
| lock_wait_time | xs:nonNegativeInteger | lock_wait_time - Time waited on locks |
| lock_wait_time_global | xs:long | lock_wait_time_global - Lock wait time global monitor element |
| lock_waits | xs:nonNegativeInteger | lock_waits - Lock waits |
| lock_waits_global | xs:long | lock_waits_global - Lock waits global monitor element |
| log_buffer_wait_time | xs:nonNegativeInteger | log_buffer_wait_time - Log buffer wait time |
| log_disk_wait_time | xs:nonNegativeInteger | log_disk_wait_time - Log disk wait time |
| log_disk_waits_total | xs:nonNegativeInteger | log_disk_waits_total - Log disk waits total |
| member | xs:nonNegativeInteger | member - Database member |
| merge_sql_stmts | xs:long | merge_sql_stmts - Merge SQL statements executed monitor element |
| num_log_buffer_full | xs:nonNegativeInteger | num_log_buffer_full - Number of full log buffers |
| num_lw_thresh_exceeded | xs:nonNegativeInteger | num_lw_thresh_exceeded - Number of thresholds exceeded |
| olap_func_overflows | xs:long | olap_func_overflows - OLAP Function Overflows monitor element |
| pkg_cache_inserts | xs:nonNegativeInteger | pkg_cache_inserts - Package cache inserts |
| pkg_cache_lookups | xs:nonNegativeInteger | pkg_cache_lookups - Package cache lookups |
| pool_col_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |

Table 159. Detailed metrics returned for MON_GET_SERVICE_SUBCLASS_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|---|-----------------------|--|
| pool_col_gbp_invalid_pages | xs:nonNegativeInteger | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| pool_col_gbp_l_reads | xs:nonNegativeInteger | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| pool_col_gbp_p_reads | xs:nonNegativeInteger | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| pool_col_l_reads | xs:nonNegativeInteger | pool_col_l_reads - Buffer pool column-organized logical reads |
| pool_col_lbp_pages_found | xs:nonNegativeInteger | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found |
| pool_col_p_reads | xs:nonNegativeInteger | pool_col_p_reads - Buffer pool column-organized physical reads |
| pool_col_writes | xs:nonNegativeInteger | pool_col_writes - Buffer pool column-organized writes |
| pool_data_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| pool_data_gbp_invalid_pages | xs:long | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages monitor element |
| pool_data_gbp_l_reads | xs:long | pool_data_gbp_l_reads - Group buffer pool data logical reads monitor element |
| pool_data_gbp_p_reads | xs:long | pool_data_gbp_p_reads - Group buffer pool data physical reads monitor element |
| pool_data_l_reads | xs:nonNegativeInteger | pool_data_l_reads - Buffer pool data logical reads |
| pool_data_lbp_pages_found | xs:long | pool_data_lbp_pages_found - Local buffer pool found data pages monitor element |
| pool_data_p_reads | xs:nonNegativeInteger | pool_data_p_reads - Buffer pool data physical reads |
| pool_data_writes | xs:nonNegativeInteger | pool_data_writes - Buffer pool data writes |
| pool_failed_async_col_reqs | xs:nonNegativeInteger | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| pool_failed_async_data_reqs | xs:nonNegativeInteger | pool_failed_async_data_reqs - Failed data prefetch requests monitor element |
| pool_failed_async_index_reqs | xs:nonNegativeInteger | pool_failed_async_index_reqs - Failed index prefetch requests monitor element |
| pool_failed_async_temp_col_reqs | xs:nonNegativeInteger | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |
| pool_failed_async_temp_data_reqs | xs:nonNegativeInteger | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element |
| pool_failed_async_temp_index_reqs | xs:nonNegativeInteger | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element |
| pool_failed_async_temp_xda_reqs | xs:nonNegativeInteger | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element |
| pool_failed_async_other_reqs | xs:nonNegativeInteger | pool_failed_async_other_reqs - Failed non-prefetch requests monitor element |
| pool_failed_async_xda_reqs | xs:nonNegativeInteger | pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element |
| pool_index_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| pool_index_gbp_invalid_pages | xs:long | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages monitor element |
| pool_index_gbp_l_reads | xs:long | pool_index_gbp_l_reads - Group buffer pool index logical reads monitor element |
| pool_index_gbp_p_reads | xs:long | pool_index_gbp_p_reads - Group buffer pool index physical reads monitor elements |

Table 159. Detailed metrics returned for MON_GET_SERVICE_SUBCLASS_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|------------------------------------|-----------------------|--|
| pool_index_l_reads | xs:nonNegativeInteger | pool_index_l_reads - Buffer pool index logical reads |
| pool_index_lbp_pages_found | xs:long | pool_index_lbp_pages_found - Local buffer pool index pages found monitor element |
| pool_index_p_reads | xs:nonNegativeInteger | pool_index_p_reads - Buffer pool index physical reads |
| pool_index_writes | xs:nonNegativeInteger | pool_index_writes - Buffer pool index writes |
| pool_queued_async_col_reqs | xs:nonNegativeInteger | pool_queued_async_col_reqs - Column-organized prefetch requests |
| pool_queued_async_data_pages | xs:nonNegativeInteger | pool_queued_async_data_pages - Data pages prefetch requests monitor element |
| pool_queued_async_data_reqs | xs:nonNegativeInteger | pool_queued_async_data_reqs - Data prefetch requests monitor element |
| pool_queued_async_index_reqs | xs:nonNegativeInteger | pool_queued_async_index_reqs - Index prefetch requests monitor element |
| pool_queued_async_col_pages | xs:nonNegativeInteger | pool_queued_async_col_pages - Column-organized page prefetch requests |
| pool_queued_async_index_pages | xs:nonNegativeInteger | pool_queued_async_index_pages - Index pages prefetch requests monitor element |
| pool_queued_async_temp_col_pages | xs:nonNegativeInteger | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| pool_queued_async_temp_col_reqs | xs:nonNegativeInteger | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| pool_queued_async_temp_data_pages | xs:nonNegativeInteger | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_data_reqs | xs:nonNegativeInteger | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_index_pages | xs:nonNegativeInteger | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_index_reqs | xs:nonNegativeInteger | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_xda_pages | xs:nonNegativeInteger | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_xda_reqs | xs:nonNegativeInteger | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element |
| pool_queued_async_other_reqs | xs:nonNegativeInteger | pool_queued_async_other_reqs - Non-prefetch requests monitor element |
| pool_queued_async_xda_pages | xs:long | pool_queued_async_xda_pages - XDA pages prefetch requests monitor element |
| pool_queued_async_xda_reqs | xs:nonNegativeInteger | pool_queued_async_xda_reqs - XDA prefetch requests monitor element |
| pool_read_time | xs:nonNegativeInteger | pool_read_time - Total buffer pool physical read time |
| pool_temp_col_l_reads | xs:nonNegativeInteger | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| pool_temp_col_p_reads | xs:nonNegativeInteger | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| pool_temp_data_l_reads | xs:nonNegativeInteger | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| pool_temp_data_p_reads | xs:nonNegativeInteger | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| pool_temp_index_l_reads | xs:nonNegativeInteger | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| pool_temp_index_p_reads | xs:nonNegativeInteger | pool_temp_index_p_reads - Buffer pool temporary index physical reads |

Table 159. Detailed metrics returned for MON_GET_SERVICE_SUBCLASS_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|---------------------------------------|-----------------------|--|
| pool_temp_xda_l_reads | xs:nonNegativeInteger | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |
| pool_temp_xda_p_reads | xs:nonNegativeInteger | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| pool_write_time | xs:nonNegativeInteger | pool_write_time - Total buffer pool physical write time |
| pool_xda_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |
| pool_xda_gbp_invalid_pages | xs:nonNegativeInteger | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| pool_xda_gbp_l_reads | xs:nonNegativeInteger | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |
| pool_xda_gbp_p_reads | xs:nonNegativeInteger | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |
| pool_xda_l_reads | xs:nonNegativeInteger | pool_xda_l_reads - Buffer pool XDA data logical reads |
| pool_xda_lbp_pages_found | xs:nonNegativeInteger | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |
| pool_xda_p_reads | xs:nonNegativeInteger | pool_xda_p_reads - Buffer pool XDA data physical reads |
| pool_xda_writes | xs:nonNegativeInteger | pool_xda_writes - Buffer pool XDA data writes |
| post_shrthreshold_hash_joins | xs:long | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| post_shrthreshold_sorts | xs:nonNegativeInteger | post_shrthreshold_sorts - Post shared threshold sorts |
| post_threshold_col_vector_consumers | xs:long | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| post_threshold_hash_grpbys | xs:long | post_threshold_hash_grpbys - Hash group by threshold |
| post_threshold_hash_joins | xs:long | post_threshold_hash_joins - Hash Join Threshold monitor element |
| post_threshold_olap_funcs | xs:long | post_threshold_olap_funcs - OLAP function threshold monitor element |
| post_threshold_peas | xs:long | post_threshold_peas - Partial early aggregation threshold |
| post_threshold_peds | xs:long | post_threshold_peds - Partial early distincts threshold |
| prefetch_wait_time | xs:nonNegativeInteger | prefetch_wait_time - Time waited for prefetch |
| prefetch_waits | xs:nonNegativeInteger | prefetch_waits - Prefetcher wait count monitor element |
| fcm_tq_send_waits_total | xs:nonNegativeInteger | fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |
| post_threshold_sorts | xs:long | post_threshold_sorts - Post threshold sorts |
| reclaim_wait_time | xs:long | reclaim_wait_time - Reclaim wait time monitor element |
| rows_deleted | xs:long | rows_deleted - Rows deleted monitor element |
| rows_inserted | xs:long | rows_inserted - Rows inserted monitor element |
| rows_modified | xs:nonNegativeInteger | rows_modified - Rows modified |
| rows_read | xs:nonNegativeInteger | rows_read - Rows read |
| rows_returned | xs:nonNegativeInteger | rows_returned - Rows returned |
| rows_updated | xs:long | rows_updated - Rows updated monitor element |
| rqsts_completed_total | xs:nonNegativeInteger | rqsts_completed_total - Total requests completed |
| select_sql_stmts | xs:long | select_sql_stmts - Select SQL Statements Executed monitor element |
| service_class_id | xs:nonNegativeInteger | service_class_id - Service class ID |
| service_subclass_name | xs:string(128) | service_subclass_name - Service subclass name |
| service_superclass_name | xs:string(128) | service_superclass_name - Service superclass name |

Table 159. Detailed metrics returned for MON_GET_SERVICE_SUBCLASS_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|--|-----------------------|--|
| sort_heap_allocated | xs:long | sort_heap_allocated - Total Sort Heap Allocated |
| sort_overflows | xs:nonNegativeInteger | sort_overflows - Sort overflows |
| sort_shrheap_allocated | xs:long | sort_shrheap_allocated - Sort Share Heap Currently Allocated |
| spacemappage_reclaim_wait_time | xs:long | spacemappage_reclaim_wait_time - Space map page reclaim wait time monitor element |
| static_sql_stmts | xs:long | static_sql_stmts - Static SQL Statements Attempted monitor element |
| tcpip_rcv_volume | xs:nonNegativeInteger | tcpip_rcv_volume - TCP/IP received volume |
| tcpip_rcv_wait_time | xs:nonNegativeInteger | tcpip_rcv_wait_time - TCP/IP rcv wait time |
| tcpip_rcvs_total | xs:nonNegativeInteger | tcpip_rcvs_total - TCP/IP rcvs total |
| tcpip_send_volume | xs:nonNegativeInteger | tcpip_send_volume - TCP/IP send volume |
| tcpip_send_wait_time | xs:nonNegativeInteger | tcpip_send_wait_time - TCP/IP send wait time |
| tcpip_sends_total | xs:nonNegativeInteger | tcpip_sends_total - TCP/IP sends total |
| thresh_violations | xs:nonNegativeInteger | thresh_violations - Number of threshold violations |
| total_act_time | xs:nonNegativeInteger | total_act_time - Total activity time |
| total_act_wait_time | xs:nonNegativeInteger | total_act_wait_time - Total activity wait time |
| total_app_commits | xs:nonNegativeInteger | total_app_commits - Total application commits |
| total_app_rollbacks | xs:nonNegativeInteger | total_app_rollbacks - Total application rollbacks |
| total_app_rqst_time | xs:nonNegativeInteger | total_app_rqst_time - Total application request time |
| total_app_section_executions | xs:nonNegativeInteger | total_app_section_executions - Total section executions |
| total_backup_proc_time | xs:long | total_backup_proc_time - Total non-wait time for online backups |
| total_backup_time | xs:long | total_backup_time - Total elapsed time for online backups |
| total_backups | xs:long | total_backups - Total backups |
| total_col_executions | xs:nonNegativeInteger | total_col_executions - Total column-organized executions |
| total_col_proc_time | xs:nonNegativeInteger | total_col_proc_time - Total column-organized processing time |
| total_col_synopsis_executions | xs:nonNegativeInteger | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |
| total_col_synopsis_proc_time | xs:nonNegativeInteger | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| total_col_synopsis_time | xs:nonNegativeInteger | total_col_synopsis_time - Total column-organized synopsis table time monitor element |
| total_col_time | xs:nonNegativeInteger | total_col_time - Total column-organized time |
| total_col_vector_consumers | xs:long | total_col_vector_consumers - Total columnar_vector consumers |
| total_commit_proc_time | xs:nonNegativeInteger | total_commit_proc_time - Total commits processing time |
| total_commit_time | xs:nonNegativeInteger | total_commit_time - Total commit time |
| total_compilations | xs:nonNegativeInteger | total_compilations - Total compilations |
| total_compile_proc_time | xs:nonNegativeInteger | total_compile_proc_time - Total compile processing time |
| total_compile_time | xs:nonNegativeInteger | total_compile_time - Total compile time |
| total_connect_authentications | xs:nonNegativeInteger | total_connect_authentications - Connections or switch user authentications performed |
| total_connect_authentication_proc_time | xs:nonNegativeInteger | total_connect_authentication_proc_time - Total connection authentication processing time |
| total_connect_authentication_time | xs:nonNegativeInteger | total_connect_authentication_time - Total connection or switch user authentication request time |
| total_connect_request_proc_time | xs:nonNegativeInteger | total_connect_request_proc_time - Total connection or switch user request processing time |
| total_connect_request_time | xs:nonNegativeInteger | total_connect_request_time - Total connection or switch user request time |

Table 159. Detailed metrics returned for MON_GET_SERVICE_SUBCLASS_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|-----------------------------------|-----------------------|--|
| total_connect_requests | xs:nonNegativeInteger | total_connect_requests - Connection or switch user requests |
| total_cpu_time | xs:nonNegativeInteger | total_cpu_time - Total CPU time |
| total_disp_run_queue_time | xs:long | total_disp_run_queue_time - Total dispatcher run queue time |
| total_extended_latch_wait_time | xs:nonNegativeInteger | total_extended_latch_wait_time - Total extended latch wait time |
| total_extended_latch_waits | xs:nonNegativeInteger | total_extended_latch_waits - Total extended latch waits |
| total_hash_grpbys | xs:long | total_hash_grpbys - Total hash group by operations |
| total_hash_joins | xs:long | total_hash_joins - Total Hash Joins monitor element |
| total_hash_loops | xs:long | total_hash_loops - Total Hash Loops monitor element |
| total_implicit_compilations | xs:nonNegativeInteger | total_implicit_compilations - Total implicit complications |
| total_implicit_compile_proc_time | xs:nonNegativeInteger | total_implicit_compile_proc_time - Total implicit compile processing time |
| total_implicit_compile_time | xs:nonNegativeInteger | total_implicit_compile_time - Total implicit compile time |
| total_index_build_proc_time | xs:long | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation, recreation or rebuilding |
| total_index_build_time | xs:long | total_index_build_time - Total time spent building indexes due to index creation, recreation or rebuilding |
| total_indexes_built | xs:long | total_indexes_built - Total number of indexes built |
| total_load_proc_time | xs:nonNegativeInteger | total_load_proc_time - Total load processing time |
| total_load_time | xs:nonNegativeInteger | total_load_time - Total load time |
| total_loads | xs:nonNegativeInteger | total_loads - Total loads |
| total_olap_funcs | xs:long | total_olap_funcs - Total OLAP Functions monitor element |
| total_peas | xs:long | total_peas - Total partial early aggregations |
| total_peds | xs:long | total_peds - Total partial early distincts |
| total_reorg_proc_time | xs:nonNegativeInteger | total_reorg_proc_time - Total reorganization processing time |
| total_reorg_time | xs:nonNegativeInteger | total_reorg_time - Total reorganization time |
| total_reorgs | xs:nonNegativeInteger | total_reorgs - Total reorganizations |
| total_rollback_proc_time | xs:nonNegativeInteger | total_rollback_proc_time - Total rollback processing time |
| total_rollback_time | xs:nonNegativeInteger | total_rollback_time - Total rollback time |
| total_routine_invocations | xs:nonNegativeInteger | total_routine_invocations - Total routine invocations |
| total_routine_time | xs:nonNegativeInteger | total_routine_time - Total routine time |
| total_routine_user_code_proc_time | xs:nonNegativeInteger | total_routine_user_code_proc_time - Total routine user code processing time |
| total_routine_user_code_time | xs:nonNegativeInteger | total_routine_user_code_time - Total routine user code time |
| total_rqst_mapped_in | xs:nonNegativeInteger | total_rqst_mapped_in - Total request mapped-in |
| total_rqst_mapped_out | xs:nonNegativeInteger | total_rqst_mapped_out - Total request mapped-out |
| total_rqst_time | xs:nonNegativeInteger | total_rqst_time - Total request time |
| total_runstats | xs:nonNegativeInteger | total_runstats - Total runtime statistics |
| total_runstats_proc_time | xs:nonNegativeInteger | total_runstats_proc_time - Total runtime statistics processing time |
| total_runstats_time | xs:nonNegativeInteger | total_runstats_time - Total runtime statistics |
| total_section_proc_time | xs:nonNegativeInteger | total_section_proc_time - Total section processing time |
| total_section_sort_proc_time | xs:nonNegativeInteger | total_section_sort_proc_time - Total section sort processing time |
| total_section_sort_time | xs:nonNegativeInteger | total_section_sort_time - Total section sort time |
| total_section_sorts | xs:nonNegativeInteger | total_section_sorts - Total section sorts |
| total_section_time | xs:nonNegativeInteger | total_section_time - Total section time |
| total_sorts | xs:nonNegativeInteger | total_sorts - Total Sorts |

Table 159. Detailed metrics returned for MON_GET_SERVICE_SUBCLASS_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|-----------------------------------|-----------------------|--|
| total_stats_fabrication_proc_time | xs:nonNegativeInteger | total_stats_fabrication_proc_time - Total statistics fabrication processing time |
| total_stats_fabrication_time | xs:nonNegativeInteger | total_stats_fabrication_time - Total statistics fabrication time |
| total_stats_fabrications | xs:nonNegativeInteger | total_stats_fabrications - Total statistics fabrications |
| total_sync_runstats | xs:nonNegativeInteger | total_sync_runstats - Total synchronous RUNSTATS activities |
| total_sync_runstats_proc_time | xs:nonNegativeInteger | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time |
| total_sync_runstats_time | xs:nonNegativeInteger | total_sync_runstats_time - Total synchronous RUNSTATS time |
| total_wait_time | xs:nonNegativeInteger | total_wait_time - Total wait time |
| tq_sort_heap_rejections | xs:long | tq_sort_heap_rejections - Table queue sort heap rejections |
| tq_sort_heap_requests | xs:long | tq_sort_heap_requests - Table queue sort heap requests |
| tq_tot_send_spills | xs:nonNegativeInteger | tq_tot_send_spills - Total number of table queue buffers overflowed |
| uid_sql_stmts | xs:long | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| wlm_queue_assignments_total | xs:nonNegativeInteger | wlm_queue_assignments_total - Workload manager total queue assignments |
| wlm_queue_time_total | xs:nonNegativeInteger | wlm_queue_time_total - Workload manager total queue time |
| xquery_stmts | xs:long | xquery_stmts - XQuery Statements Attempted monitor element |

Usage notes

The metrics returned by the MON_GET_SERVICE_SUBCLASS_DETAILS table function represent the accumulation of all metrics for requests that have executed under the indicated service subclass. This function is similar to the MON_GET_SERVICE_SUBCLASS table function:

- The the MON_GET_SERVICE_SUBCLASS table function returns the most commonly used metrics in a column based format and is the most performance efficient method of retrieving metrics.
- The MON_GET_SERVICE_SUBCLASS_DETAILS table function returns the entire set of available metrics in an XML document format, which provides maximum flexibility for formatting output. The XML-based output can be parsed directly by an XML parser, or it can be converted to relational format by the XMLTABLE function (see the example).

Metrics are rolled up to a service class on unit of work boundaries, and periodically during the execution of requests. Therefore, the values reported by this table function reflect the current state of the system at the time of the most recent rollup. Metrics are strictly increasing in value. To determine the value of a given metric for an interval of time, use the MON_GET_SERVICE_SUBCLASS_DETAILS table function to query the metric at the start and end of the interval, and compute the difference.

Request metrics are controlled through the COLLECT REQUEST METRICS clause on service superclasses, and the **mon_req_metrics** database configuration parameter at the database level. Metrics are only collected for a request if the request is processed by an agent in a service subclass whose parent service superclass has request metrics enabled, or if request metrics collection is enabled for the entire database. By default request metrics are enabled at the database level. If request metrics are disabled at the database level and for a service superclass, the metrics

reported for each connection mapped to that service superclass stop increasing (or remain at 0 if request metrics were disabled at database activation time).

The MON_GET_SERVICE_SUBCLASS_DETAILS table function returns one row of data per service subclass and per member. No aggregation across service classes (on a member), or across members (for a service class or more), is performed. However, aggregation can be achieved through SQL queries (see the example). The input parameters have the effect of being ANDed together. Therefore, if you specify conflicting input parameters (for example, a superclass name SUPA and subclass name SUBB that is not a subclass of SUPA), no rows are returned.

In the data returned by this function that describe the default subclass SYSDEFAULTSUBCLASS under the superclass SYSDEFAULTSYSTEMCLASS, all metrics have a value of 0.

Tip: A request might execute in more than one service subclass. For example, this situation might occur if a request is mapped from one service subclass to another by using a Workload Manager (WLM) threshold with a REMAP ACTIVITY action. Although the time spent metrics are updated for each service subclass under which the request executes, the request counters are incremented for the service subclass where the request completed. Therefore, you should not analyze the averages of request times for a single subclass. All subclasses to which an activity can be mapped must be analyzed in conjunction with one another. For example, if a threshold exists that can map activities from service subclass A to service subclass B, then when you compute averages of requests, you should aggregate the counters and metrics for service subclasses A and B, and compute the averages using the aggregates.

Note that the metrics from background EDUs such as prefetchers and page cleaners are not rolled up to service classes.

The schema for the XML document that is returned in the DETAILS column is available in the file sqllib/misc/DB2MonRoutines.xsd. Further details can be found in the file sqllib/misc/DB2MonCommon.xsd.

Example

Display the total CPU time used and total number of requests processed for each service superclass, ordered by CPU usage in relational format (using XMLTABLE).

```
SELECT varchar(scmetrics.service_superclass_name,30) as service_superclass,
       sum(detmetrics.total_cpu_time) as total_cpu,
       sum(detmetrics.app_rqsts_completed_total) as total_rqsts
FROM TABLE(MON_GET_SERVICE_SUBCLASS_DETAILS('','',-2)) AS SCMETRICS,
XMLTABLE (XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'),
          '$detmetric/db2_service_subclass'
          PASSING XMLPARSE(DOCUMENT SCMETRICS.DETAILS)
          as "detmetric"
COLUMNS "TOTAL_CPU_TIME" INTEGER PATH 'system_metrics/total_cpu_time',
         "APP_RQSTS_COMPLETED_TOTAL" INTEGER PATH
         'system_metrics/app_rqsts_completed_total')
AS DETMETRICS
GROUP BY service_superclass_name
ORDER BY total_cpu desc
```

The following is an example of output from this query.

| SERVICE_SUPERCLASS | TOTAL_CPU | TOTAL_RQSTS |
|---------------------|-----------|-------------|
| SYSDEFAULTUSERCLASS | 2428188 | 26 |

| | | |
|----------------------------|---|---|
| SYSDEFAULTMAINTENANCECLASS | 0 | 0 |
| SYSDEFAULTSYSTEMCLASS | 0 | 0 |

3 record(s) selected.

MON_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses

The MON_GET_SERVICE_SUBCLASS_STATS function returns basic statistics for one or more service subclasses.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Syntax

```

►►MON_GET_SERVICE_SUBCLASS_STATS(—service_superclass_name—,—————►
►service_subclass_name—,—member—)—————►►

```

The schema is SYSPROC.

Routine parameters

service_superclass_name

An input argument of type VARCHAR(128) that specifies the name of a service superclass in the currently connected database. If the argument is null or an empty string, the data is retrieved for all of the superclasses in the database.

service_subclass_name

An input argument of type VARCHAR(128) that specifies the name of a service subclass in the currently connected database. If the argument is null or an empty string, the data is retrieved for all of the subclasses in the database.

member

An input argument of type INTEGER that specifies a valid member number in the same instance as the currently connected database. Specify -1 for the current member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

Information returned

Table 160. Information returned by the MON_GET_SERVICE_SUBCLASS_STATS routine

| Column name | Data type | Description |
|-------------------------|--------------|---|
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name monitor element |

Table 160. Information returned by the MON_GET_SERVICE_SUBCLASS_STATS routine (continued)

| Column name | Data type | Description |
|---------------------------|--------------|---|
| SERVICE_SUBCLASS_NAME | VARCHAR(128) | service_subclass_name - Service subclass name monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |
| COORD_ACT_COMPLETED_TOTAL | BIGINT | coord_act_completed_total - Coordinator activities completed total monitor element |
| COORD_ACT_ABORTED_TOTAL | BIGINT | coord_act_aborted_total - Coordinator activities aborted total monitor element |
| COORD_ACT_REJECTED_TOTAL | BIGINT | coord_act_rejected_total - Coordinator activities rejected total monitor element |
| CONCURRENT_ACT_TOP | INTEGER | concurrent_act_top - Concurrent activity top monitor element |
| COORD_ACT_LIFETIME_TOP | BIGINT | coord_act_lifetime_top - Coordinator activity lifetime top monitor element |
| COORD_ACT_LIFETIME_AVG | DOUBLE | coord_act_lifetime_avg - Coordinator activity lifetime average monitor element |
| COORD_ACT_LIFETIME_STDDEV | DOUBLE | <p>Standard deviation of lifetime for coordinator activities at nesting level 0 that were associated with this service subclass since the last reset. If the COLLECT AGGREGATE ACTIVITY DATA parameter of the service class is set to NONE, the value of the column is null. Units are milliseconds.</p> <p>This standard deviation is computed from the coordinator activity lifetime histogram and may be inaccurate if the histogram is not correctly sized to fit the data. The value of -1 is returned if any values fall into the last histogram bin.</p> <p>The COORD_ACT_LIFETIME_STDDEV value of a service subclass is unaffected by activities that pass through the service subclass but are remapped to a different subclass before they are completed.</p> |
| COORD_ACT_EXEC_TIME_AVG | DOUBLE | coord_act_exec_time_avg - Coordinator activities execution time average monitor element |

Table 160. Information returned by the MON_GET_SERVICE_SUBCLASS_STATS routine (continued)

| Column name | Data type | Description |
|-----------------------------|-----------|--|
| COORD_ACT_EXEC_TIME_STDDEV | DOUBLE | <p>Standard deviation of the execution times for coordinator activities at nesting level 0 that were associated with this service subclass since the last reset. Units are milliseconds.</p> <p>This standard deviation is computed from the coordinator activity executetime histogram and may be inaccurate if the histogram is not correctly sized to fit the data. The value of -1 is returned if any values fall into the last histogram bin.</p> <p>The execution time standard deviation of a service subclass is unaffected by activities that pass through the subclass but are remapped to a different subclass before they are completed.</p> |
| COORD_ACT_QUEUE_TIME_AVG | DOUBLE | coord_act_queue_time_avg - Coordinator activity queue time average monitor element |
| COORD_ACT_QUEUE_TIME_STDDEV | DOUBLE | <p>Standard deviation of the queue time for coordinator activities at nesting level 0 that were associated with this service subclass since the last reset. If the COLLECT AGGREGATE ACTIVITY DATA parameter of the service class is set to NONE, the value of the column is null. Units are milliseconds.</p> <p>This standard deviation is computed from the coordinator activity queue time histogram and may be inaccurate if the histogram is not correctly sized to fit the data. The value of -1 is returned if any values fall into the last histogram bin.</p> <p>The queue time standard deviation is counted only toward the service subclass in which the activity was queued.</p> |
| NUM_REQUESTS_ACTIVE | BIGINT | Number of requests that are running in the service subclass at the time that this table function is running. |
| NUM_REQUESTS_TOTAL | BIGINT | <p>Number of requests that finished running in this service subclass since the last reset. This finished state applies to any request regardless of its membership in an activity. If the COLLECT AGGREGATE ACTIVITY DATA parameter of the service class is set to NONE, the value of the column is null.</p> <p>The NUM_REQUESTS_TOTAL value of a service subclass is unaffected by requests that pass through the service subclass but are not completed in it.</p> |

Table 160. Information returned by the MON_GET_SERVICE_SUBCLASS_STATS routine (continued)

| Column name | Data type | Description |
|--------------------------|-----------|---|
| REQUEST_EXEC_TIME_AVG | DOUBLE | request_exec_time_avg - Request execution time average monitor element |
| REQUEST_EXEC_TIME_STDDEV | DOUBLE | <p>Standard deviation of the execution times for requests that were associated with this service subclass since the last reset. Units are milliseconds. If the COLLECT AGGREGATE REQUEST DATA parameter of the service class is set to NONE, the value of this column is NULL.</p> <p>This standard deviation is computed from the request executetime histogram and may be inaccurate if the histogram is not correctly sized to fit the data. The value of -1 is returned if any values fall into the last histogram bin.</p> <p>The execution time standard deviation of a service subclass is unaffected by requests that pass through the subclass but were not completed in it.</p> |
| REQUEST_EXEC_TIME_TOTAL | BIGINT | <p>Sum of the execution times for requests that were associated with this service subclass since the last reset. Units are milliseconds. If the COLLECT AGGREGATE REQUEST DATA parameter of the service class is set to NONE, the value of this column is NULL.</p> <p>This total is computed from the request execution time histogram and may be inaccurate if the histogram is not correctly sized to fit the data. The value of -1 is returned if any values fall into the last histogram bin.</p> <p>The execution time total of a service subclass is unaffected by requests that pass through the subclass but are not completed in it.</p> |
| ACT_REMAPPED_IN | BIGINT | act_remapped_in - Activities remapped in monitor element |
| ACT_REMAPPED_OUT | BIGINT | act_remapped_out - Activities remapped out monitor element |
| CONCURRENT_WLO_TOP | INTEGER | concurrent_wlo_top - Concurrent workload occurrences top monitor element |
| UOW_TOTAL_TIME_TOP | BIGINT | uow_total_time_top - UOW total time top monitor element |
| UOW_THROUGHPUT | DOUBLE | <p>uow_throughput - Unit of work throughput monitor element</p> <p>The unit of work throughput since the last reset of the statistics.</p> |
| UOW_LIFETIME_AVG | DOUBLE | uow_lifetime_avg - Unit of work lifetime average monitor element |

Table 160. Information returned by the MON_GET_SERVICE_SUBCLASS_STATS routine (continued)

| Column name | Data type | Description |
|---------------------------------|-----------|--|
| UOW_COMPLETED_TOTAL | BIGINT | uow_completed_total - Total completed units of work monitor element |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time monitor element |
| TOTAL_DISP_RUN_QUEUE_TIME | BIGINT | total_disp_run_queue_time - Total dispatcher run queue time monitor element |
| ACT_THROUGHPUT | DOUBLE | act_throughput - Activity throughput monitor element |
| CPU_UTILIZATION | DOUBLE | cpu_utilization - CPU utilization monitor element |
| APP_ACT_COMPLETED_TOTAL | BIGINT | app_act_completed_total - Total successful external coordinator activities monitor element |
| APP_ACT_ABORTED_TOTAL | BIGINT | app_act_aborted_total - Total failed external coordinator activities monitor element |
| APP_ACT_REJECTED_TOTAL | BIGINT | app_act_rejected_total - Total rejected external coordinator activities monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| ACTIVE_HASH_GRPBYS_TOP | BIGINT | active_hash_grpbys_top - Active hash Group By operations high watermark |
| ACTIVE_HASH_JOINS_TOP | BIGINT | active_hash_joins_top - Active hash joins operations high watermark |
| ACTIVE_OLAP_FUNCS_TOP | BIGINT | active_olap_funcs_top - Active OLAP functions operations high watermark |
| ACTIVE_PEAS_TOP | BIGINT | active_peas_top - Active partial early aggregations operations high watermark |
| ACTIVE_PEDS_TOP | BIGINT | active_peds_top - Active partial early distinct operations high watermark |
| ACTIVE_SORT_CONSUMERS_TOP | BIGINT | active_sort_consumers_top - Active sort consumers high watermark |
| ACTIVE_SORTS_TOP | BIGINT | active_sorts_top - Active Sorts high watermark |
| ACTIVE_COL_VECTOR_CONSUMERS_TOP | BIGINT | active_col_vector_consumers_top - Active columnar_vector consumers high watermark |
| SORT_CONSUMER_HEAP_TOP | BIGINT | sort_consumer_heap_top - Individual private sort heap consumer high watermark |
| SORT_CONSUMER_SHRHEAP_TOP | BIGINT | sort_consumer_shrheap_top - Individual shared sort heap consumer high watermark |
| SORT_HEAP_TOP | BIGINT | sort_heap_top - Sort private heap high watermark |
| SORT_SHRHEAP_TOP | BIGINT | sort_shrheap_top - Sort share heap high watermark |

Usage notes

Some statistics are returned only if you set the COLLECT AGGREGATE ACTIVITY DATA and COLLECT AGGREGATE REQUEST DATA parameters for the corresponding service subclass to a value other than NONE.

The MON_GET_SERVICE_SUBCLASS_STATS table function returns one row of data per service subclass and per member. The function does not aggregate data across service classes (on a partition) or across partitions (for one or more service classes). However, you can use SQL queries to aggregate data.

The parameters are assessed in conjunction with each other. If you specify conflicting input parameters, no rows are returned. For example, if you specify a superclass named SUPA and a subclass named SUBB (where SUBB is not a subclass of SUPA), no rows are returned.

Examples

Example 1: Because every activity must be mapped to a database service class before being run, you can monitor the global state of the system by using the service class statistics table functions and querying all of the service classes on all members. In the following example, a null value is passed for *service_superclass_name* and *service_subclass_name* to return statistics for all service classes, and the value -2 is specified for *member* to return statistics for all members:

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       CAST(COORD_ACT_LIFETIME_AVG / 1000 AS DECIMAL(9,3))
       AS AVGLIFETIME,
       CAST(COORD_ACT_LIFETIME_STDDEV / 1000 AS DECIMAL(9,3))
       AS STDDEVLIFETIME,
       SUBSTR(CAST(LAST_RESET AS VARCHAR(30)),1,16) AS LAST_RESET
FROM TABLE(MON_GET_SERVICE_SUBCLASS_STATS(CAST(NULL AS VARCHAR(128)),
      CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, MEMB
```

The statement returns service class statistics such as average activity lifetime and standard deviation in seconds, as shown in the following sample output:

| SUPERCLASS_NAME | SUBCLASS_NAME | MEMB | ... |
|---------------------|--------------------|----------------|------------------|
| ----- | ----- | ---- | ... |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 0 | ... |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 1 | ... |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 2 | ... |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 3 | ... |
| ... | AVGLIFETIME | STDDEVLIFETIME | LAST_RESET |
| ... | ----- | ----- | ----- |
| ... | 691.242 | 34.322 | 2006-07-24-11.44 |
| ... | 644.740 | 22.124 | 2006-07-24-11.44 |
| ... | 612.431 | 43.347 | 2006-07-24-11.44 |
| ... | 593.451 | 28.329 | 2006-07-24-11.44 |

Example 2: The same table function can also give the highest value for average concurrency of coordinator activities running in the service class on each member:

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       CONCURRENT_ACT_TOP AS ACTTOP,
       CONCURRENT_WLO_TOP AS CONNTOP
```

```
FROM TABLE(MON_GET_SERVICE_SUBCLASS_STATS(CAST(NULL AS VARCHAR(128)),
      CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, MEMB
```

Sample output is as follows:

| SUPERCLASS_NAME | SUBCLASS_NAME | MEMB | ACTTOP | CONNTOP |
|---------------------|----------------------|------|--------|---------|
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS 0 | | 10 | 7 |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS 1 | | 0 | 0 |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS 2 | | 0 | 0 |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS 3 | | 0 | 0 |

By checking the average execution times and numbers of activities in the output of this table function, you can get a good high-level view of the load on each member for a specific database. Any significant variations in the high-level gauges returned by this table function might indicate a change in the load on the system.

Example 3: If an activity uses thresholds with REMAP ACTIVITY TO actions, the activity might spend time in more than one service class during its lifetime. You can determine how many activities have passed through a set of service classes by looking at the ACT_REMAPPED_IN and ACT_REMAPPED_OUT columns, as shown in the following example:

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
      SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
      ACT_REMAPPED_IN AS MAPPED_IN,
      ACT_REMAPPED_OUT AS MAPPED_OUT
FROM TABLE(MON_GET_SERVICE_SUBCLASS_STATS(CAST(NULL AS VARCHAR(128)),
      CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME
```

Sample output is as follows:

| SUPERCLASS_NAME | SUBCLASS_NAME | MAPPED_IN | MAPPED_OUT |
|---------------------|--------------------|-----------|------------|
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 0 | 0 |
| SUPERCLASS1 | SYSDEFAULTSUBCLASS | 0 | 0 |
| SUPERCLASS1 | SUBCLASS1 | 0 | 7 |
| SUPERCLASS1 | SUBCLASS2 | 7 | 0 |

MON_GET_SERVICE_SUPERCLASS_STATS - Return statistics of service superclasses

The MON_GET_SERVICE_SUPERCLASS_STATS function returns basic statistics of one or more service superclasses.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Syntax

►►MON_GET_SERVICE_SUPERCLASS_STATS(—*service_superclass_name*—,—*member*—)►►

The schema is SYSPROC.

Routine parameters

service_superclass_name

An input argument of type VARCHAR(128) that specifies the name of a service superclass in the currently connected database. If the argument is null or an empty string, data is retrieved for all the superclasses in the database.

member

An input argument of type INTEGER that specifies a valid member number in the same instance as the currently connected database. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

Information returned

Table 161. Information returned by the MON_GET_SERVICE_SUPERCLASS_STATS routine

| Column name | Data type | Description |
|---------------------------|--------------|---|
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |
| CONCURRENT_CONNECTION_TOP | INTEGER | concurrent_connection_top - Concurrent connection top monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

Usage notes

The MON_GET_SERVICE_SUPERCLASS_STATS table function returns one row of data per service superclass and per member. The function does not aggregate data across service superclasses (on a member) or across members (for one or more service superclasses). However, you can use SQL queries to aggregate data, as shown in the previous example.

Example

The following query displays the basic statistics for all the service superclasses on the system, across all database members:

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME, 1, 26) SERVICE_SUPERCLASS_NAME,  
       MEMBER,  
       LAST_RESET,  
       CONCURRENT_CONNECTION_TOP CONCURRENT_CONN_TOP  
FROM TABLE(MON_GET_SERVICE_SUPERCLASS_STATS('', -2)) as SCSTATS
```

Sample output is as follows:

| SERVICE_SUPERCLASS_NAME | MEMBER | ... |
|--------------------------------|---------------------|-----|
| SYSDEFAULTSYSTEMCLASS | 0 | ... |
| SYSDEFAULTMAINTENANCECLASS | 0 | ... |
| SYSDEFAULTUSERCLASS | 0 | ... |
| ... LAST_RESET | CONCURRENT_CONN_TOP | |
| ... | ... | |
| ... 2006-09-05-09.38.44.396788 | 0 | |
| ... 2006-09-05-09.38.44.396795 | 0 | |
| ... 2006-09-05-09.38.44.396796 | 1 | |

MON_GET_TABLE table function - get table metrics

The MON_GET_TABLE table function returns monitor metrics for one or more tables.

Syntax

►► MON_GET_TABLE (—*tabschema*—, —*tablename*—, —*member*—) ◀◀

The schema is SYSPROC.

Table function parameters

tabschema

An input argument of type VARCHAR(128) that specifies a valid table schema name in the currently connected database when calling this function. If the argument is null or an empty string, metrics are retrieved for all tables in all schemas in the database. If the argument is specified, metrics are only returned for tables in the specified schema.

tablename

An input argument of type VARCHAR(128) that specifies a valid table name in the currently connected database when calling this function. If the argument is null or an empty string, metrics are retrieved for all the tables in the database.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Information returned

Table 162. Information returned for MON_GET_TABLE

| Column Name | Data Type | Description |
|------------------------------------|--------------|--|
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TABNAME | VARCHAR(128) | table_name - Table name |
| MEMBER | SMALLINT | member - Database member |
| TAB_TYPE | VARCHAR(14) | table_type - Table type |
| TAB_FILE_ID | BIGINT | table_file_id - Table file ID |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data partition identifier |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| INDEX_TBSP_ID | BIGINT | index_tbsp_id - Index table space ID |
| LONG_TBSP_ID | BIGINT | long_tbsp_id - Long table space ID |
| TABLE_SCANS | BIGINT | table_scans - Table scans |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted |
| OVERFLOW_ACCESSES | BIGINT | overflow_accesses - Accesses to overflowed records |
| OVERFLOW_CREATES | BIGINT | overflow_creates - Overflow creates |
| PAGE_REORGS | BIGINT | page_reorgs - Page reorganizations |
| DATA_OBJECT_L_PAGES | BIGINT | data_object_l_pages - Logical data object pages |
| LOB_OBJECT_L_PAGES | BIGINT | lob_object_l_pages - Logical LOB object pages |
| LONG_OBJECT_L_PAGES | BIGINT | long_object_l_pages - Logical Long object pages |
| INDEX_OBJECT_L_PAGES | BIGINT | index_object_l_pages - Logical index object pages |
| XDA_OBJECT_L_PAGES | BIGINT | xda_object_l_pages - Logical XDA object pages |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number |
| NO_CHANGE_UPDATES | BIGINT | no_change_updates - Number of no change row updates |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_WAIT_TIME_GLOBAL | BIGINT | lock_wait_time_global - Lock wait time global |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOCK_WAITS_GLOBAL | BIGINT | lock_waits_global - Lock waits global |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| LOCK_ESCALS_GLOBAL | BIGINT | lock_escals_global - Number of global lock escalations |
| DATA_SHARING_STATE | VARCHAR(19) | data_sharing_state - Data sharing state monitor element |
| DATA_SHARING_STATE_CHANGE_TIME | TIMESTAMP | data_sharing_state_change_time - Data sharing state change time monitor element |
| DATA_SHARING_REMOTE_LOCKWAIT_COUNT | BIGINT | data_sharing_remote_lockwait_count - Data sharing remote lock wait count monitor element |
| DATA_SHARING_REMOTE_LOCKWAIT_TIME | BIGINT | data_sharing_remote_lockwait_time - Data sharing remote lock wait time monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |

Table 162. Information returned for MON_GET_TABLE (continued)

| Column Name | Data Type | Description |
|--|-----------|--|
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| OBJECT_DATA_L_READS | BIGINT | object_data_l_reads - Buffer pool data logical reads for a table |
| OBJECT_DATA_P_READS | BIGINT | object_data_p_reads - Buffer pool data physical reads for a table |
| OBJECT_DATA_GBP_L_READS | BIGINT | object_data_gbp_l_reads - Group buffer pool data logical reads for a table |
| OBJECT_DATA_GBP_P_READS | BIGINT | object_data_gbp_p_reads - Group buffer pool data physical reads for a table |
| OBJECT_DATA_GBP_INVALID_PAGES | BIGINT | object_data_gbp_invalid_pages - Group buffer pool invalid data pages for a table |
| OBJECT_DATA_LBP_PAGES_FOUND | BIGINT | object_data_lbp_pages_found - Local buffer pool found data pages for a table |
| OBJECT_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | object_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool |
| OBJECT_XDA_L_READS | BIGINT | object_xda_l_reads - Buffer pool XDA data logical reads for a table |
| OBJECT_XDA_P_READS | BIGINT | object_xda_p_reads - Buffer pool XDA data physical reads for a table |
| OBJECT_XDA_GBP_L_READS | BIGINT | object_xda_gbp_l_reads - Group buffer pool XDA data logical read requests for a table |
| OBJECT_XDA_GBP_P_READS | BIGINT | object_xda_gbp_p_reads - Group buffer pool XDA data physical read requests for a table |
| OBJECT_XDA_GBP_INVALID_PAGES | BIGINT | object_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages for a table |
| OBJECT_XDA_LBP_PAGES_FOUND | BIGINT | object_xda_lbp_pages_found - Local buffer pool XDA data pages found for a table |
| OBJECT_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | object_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool |
| NUM_PAGE_DICT_BUILT | BIGINT | num_page_dict_built - Number of page-level compression dictionaries created or re-created |
| STATS_ROWS_MODIFIED | BIGINT | stats_rows_modified - Rows modified since last RUNSTATS |
| RTS_ROWS_MODIFIED | BIGINT | rts_rows_modified - Rows modified since last real time statistics |
| COL_OBJECT_L_PAGES | BIGINT | col_object_l_pages - Column-organized logical pages |
| TAB_ORGANIZATION | CHAR(1) | tab_organization - Data organization in table |
| OBJECT_COL_L_READS | BIGINT | object_col_l_reads - Column-organized logical reads |
| OBJECT_COL_P_READS | BIGINT | object_col_p_reads - Column-organized physical reads |
| OBJECT_COL_GBP_L_READS | BIGINT | object_col_gbp_l_reads - Group buffer pool column-organized logical reads |
| OBJECT_COL_GBP_P_READS | BIGINT | object_col_gbp_p_reads - Group buffer pool column-organized physical reads |
| OBJECT_COL_GBP_INVALID_PAGES | BIGINT | object_col_gbp_invalid_pages - Group buffer pool column-organized invalid data pages |
| OBJECT_COL_LBP_PAGES_FOUND | BIGINT | object_col_lbp_pages_found - Local buffer pool column-organized pages found |

Table 162. Information returned for MON_GET_TABLE (continued)

| Column Name | Data Type | Description |
|--|-----------|---|
| OBJECT_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | object_col_gbp_indep_pages_found_in_lbp - Group buffer pool column-organized index pages found in local buffer pool |
| NUM_COLUMNS_REFERENCED | BIGINT | num_columns_referenced - Number of columns referenced |
| SECTION_EXEC_WITH_COL_REFERENCES | BIGINT | section_exec_with_col_references - Section execution with column-organized references |
| OBJECT_DATA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| OBJECT_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| OBJECT_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| OBJECT_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| OBJECT_XDA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| OBJECT_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| OBJECT_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| OBJECT_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| OBJECT_COL_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| OBJECT_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| OBJECT_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| OBJECT_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| EXT_TABLE_RECV_WAIT_TIME | BIGINT | ext_table_rcv_wait_time - Total agent wait time for external table readers monitor element |
| EXT_TABLE_RECVS_TOTAL | BIGINT | ext_table_rcvs_total - Total row batches received from external table readers monitor element |
| EXT_TABLE_RECV_VOLUME | BIGINT | ext_table_rcv_volume - Total data received from external table readers monitor element |
| EXT_TABLE_READ_VOLUME | BIGINT | ext_table_read_volume - Total data read by external table readers monitor element |
| EXT_TABLE_SEND_WAIT_TIME | BIGINT | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| EXT_TABLE_SENDS_TOTAL | BIGINT | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| EXT_TABLE_SEND_VOLUME | BIGINT | ext_table_send_volume - Total data sent to external table writers monitor element |

Table 162. Information returned for MON_GET_TABLE (continued)

| Column Name | Data Type | Description |
|------------------------|-----------|---|
| EXT_TABLE_WRITE_VOLUME | BIGINT | ext_table_write_volume - Total data written by external table writers monitor element |

Usage notes

The MON_GET_TABLE table function returns one row of data per database table and per database member. If range-partitioned tables are being used, one row is returned for each table partition per database member. No aggregation across database members is performed. However, aggregation can be achieved through SQL queries as shown in the example.

Metrics are returned only for tables accessed since the database was activated.

The metrics returned by this function are collected at two levels. Some metrics are always collected, while the collection of other metrics is controlled at the database level using the mon_obj_metrics configuration parameter. Refer to the descriptions provided in the preceding table to determine if any settings must be active for data to be collected for a particular metric.

Examples

1. List the activity on all tables accessed since the database was activated, aggregated across all database members, ordered by highest number of reads.

```
SELECT varchar(tabschema,20) as tabschema,
       varchar(tabname,20) as tabname,
       sum(rows_read) as total_rows_read,
       sum(rows_inserted) as total_rows_inserted,
       sum(rows_updated) as total_rows_updated,
       sum(rows_deleted) as total_rows_deleted
FROM TABLE(MON_GET_TABLE('','",-2)) AS t
GROUP BY tabschema, tabname
ORDER BY total_rows_read DESC
```

The following is an example of output from this query.

| TABSCHEMA | TABNAME | TOTAL_ROWS_READ | ... |
|-----------|----------|-----------------|-----|
| ----- | ----- | ----- | ... |
| SYSIBM | SYSHISTO | 113 | ... |
| SYSIBM | SYSWORKL | 22 | ... |
| SYSIBM | SYSROUTI | 13 | ... |
| SYSIBM | SYSSERVI | 13 | ... |
| SYSIBM | SYSTHRES | 6 | ... |
| SYSIBM | SYSTABLE | 3 | ... |
| SYSIBM | SYSCONTE | 2 | ... |
| SYSIBM | SYSDBAUT | 2 | ... |
| SYSIBM | SYSEVENT | 2 | ... |
| SYSIBM | SYSPLAN | 1 | ... |
| SYSIBM | SYSURRO | 1 | ... |
| SYSIBM | SYSVERSI | 1 | ... |
| SYSIBM | SYSXMLST | 1 | ... |
| SYSIBM | SYSAUDIT | 0 | ... |
| SYSIBM | SYSROLEA | 0 | ... |
| SYSIBM | SYSROLES | 0 | ... |
| SYSIBM | SYSTASKS | 0 | ... |
| SYSIBM | SYSWORKA | 0 | ... |
| SYSIBM | SYSXMLPA | 0 | ... |

19 record(s) selected.

Output for query (continued).

| ... | TOTAL_ROWS_INSERTED | TOTAL_ROWS_UPDATED | TOTAL_ROWS_DELETED |
|-----|---------------------|--------------------|--------------------|
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |
| ... | 0 | 0 | 0 |

2. Get the explicit hierarchical locking state for tables in the schema JOE from member 1:

```
SELECT SUBSTR(TABNAME,1,8) AS TABNAME,
       MEMBER,
       DATA_SHARING_STATE AS DS_STATE,
       DATA_SHARING_REMOTE_LOCKWAIT_COUNT AS DSRL_COUNT,
       DATA_SHARING_REMOTE_LOCKWAIT_TIME AS DSRL_TIME
FROM   TABLE(MON_GET_TABLE('JOE','',1))
```

This query returns the following output:

| TABNAME | MEMBER | DS_STATE | DSRL_COUNT | DSRL_TIME |
|---------|--------|------------|------------|-----------|
| T1 | 1 | NOT_SHARED | 0 | 0 |
| T2 | 1 | SHARED | 2 | 21844 |
| T3 | 1 | SHARED | 0 | 0 |

This output shows that table T1 is currently in a "NOT_SHARED" state and has never been forced out of this state during the current database activation. Table T3 has never entered a "NOT_SHARED" state. Table T2 has been in a "NOT_SHARED" state twice and has had to exit both times.

MON_GET_TABLESPACE table function - Get table space metrics

The MON_GET_TABLESPACE table function returns monitor metrics for one or more table spaces.

Syntax

```
➤➤—MON_GET_TABLESPACE—(—tbsp_name—,—member—)—————➤➤
```

The schema is SYSPROC.

Table function parameters

tbsp_name

An input argument of type VARCHAR(128) that specifies a valid table space name in the currently connected database when calling this function. If the argument is null or an empty string, metrics are retrieved for all table spaces in the database.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

List table spaces ordered by number of physical reads from table space containers.

```
SELECT varchar(tbsp_name, 30) as tbsp_name,  
       member,  
       tbsp_type,  
       pool_data_p_reads  
FROM TABLE(MON_GET_TABLESPACE(' ', -2)) AS t  
ORDER BY pool_data_p_reads DESC
```

The following is an example of output from this query.

| TBSP_NAME | MEMBER | TBSP_TYPE | POOL_DATA_P_READS |
|-------------|--------|-----------|-------------------|
| SYSCATSPACE | 0 | DMS | 79 |
| USERSPACE1 | 0 | DMS | 34 |
| TEMPSPACE1 | 0 | SMS | 0 |

3 record(s) selected.

Usage notes

The MON_GET_TABLESPACE table function returns one row of data per database table space and per database member. No aggregation across database members is performed. However, aggregation can be achieved through SQL queries.

Metrics collected by this function are controlled at the database level using the mon_obj_metrics configuration parameter. By default, metrics collection is enabled.

Information returned

Table 163. Information returned for MON_GET_TABLESPACE

| Column Name | Data Type | Description or corresponding monitor element |
|-------------|--------------|--|
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| MEMBER | SMALLINT | member - Database member |

Table 163. Information returned for MON_GET_TABLESPACE (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|--------------------------|-------------|--|
| TBSP_TYPE | VARCHAR(10) | tablespace_type - Table space type. This interface returns a text identifier based on defines in sqlutil.h, and is one of: <ul style="list-style-type: none"> • DMS • SMS |
| TBSP_CONTENT_TYPE | VARCHAR(10) | tablespace_content_type - Table space content type. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • ANY • LARGE • SYSTEMP • USRTEMP |
| TBSP_PAGE_SIZE | BIGINT | tablespace_page_size - Table space page size |
| TBSP_EXTENT_SIZE | BIGINT | tablespace_extent_size - Table space extent size |
| TBSP_PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table space prefetch size |
| TBSP_CUR_POOL_ID | BIGINT | tablespace_cur_pool_id - Buffer pool currently being used |
| TBSP_NEXT_POOL_ID | BIGINT | tablespace_next_pool_id - Buffer pool that will be used at next startup |
| FS_CACHING | SMALLINT | fs_caching - File system caching |
| TBSP_REBALANCER_MODE | VARCHAR(30) | tablespace_rebalancer_mode - Rebalancer mode. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • NO_REBAL • FWD_REBAL • REV_REBAL • FWD_REBAL_OF_2PASS • REV_REBAL_OF_2PASS |
| TBSP_USING_AUTO_STORAGE | SMALLINT | tablespace_using_auto_storage - Table space enabled for automatic storage |
| TBSP_AUTO_RESIZE_ENABLED | SMALLINT | tablespace_auto_resize_enabled - Table space automatic resizing enabled |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer pool XDA data logical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |

Table 163. Information returned for MON_GET_TABLESPACE (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|----------------------------|-----------|---|
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer pool XDA data physical reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer pool XDA data writes |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer pool asynchronous XDA data reads |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer pool asynchronous XDA read requests |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer pool asynchronous XDA data writes |
| VECTORED_IOS | BIGINT | vectored_ios - Number of vectored IO requests |
| PAGES_FROM_VECTORED_IOS | BIGINT | pages_from_vectored_ios - Total number of pages read by vectored IO |
| BLOCK_IOS | BIGINT | block_ios - Number of block IO requests |
| PAGES_FROM_BLOCK_IOS | BIGINT | pages_from_block_ios - Total number of pages read by block IO |

Table 163. Information returned for MON_GET_TABLESPACE (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|-----------------------------------|--------------|---|
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| FILES_CLOSED | BIGINT | files_closed - Database files closed |
| TBSP_STATE | VARCHAR(256) | tablespace_state - Table space state |
| TBSP_USED_PAGES | BIGINT | tablespace_used_pages - Used pages in table space |
| TBSP_FREE_PAGES | BIGINT | tablespace_free_pages - Free pages in table space |
| TBSP_USABLE_PAGES | BIGINT | tablespace_usable_pages - Usable pages in table space |
| TBSP_TOTAL_PAGES | BIGINT | tablespace_total_pages - Total pages in table space |
| TBSP_PENDING_FREE_PAGES | BIGINT | tablespace_pending_free_pages - Pending free pages in table space |
| TBSP_PAGE_TOP | BIGINT | tablespace_page_top - Table space high watermark |
| TBSP_MAX_PAGE_TOP | BIGINT | tbasp_max_page_top - Maximum table space page high watermark |
| RECLAIMABLE_SPACE_ENABLED | SMALLINT | reclaimable_space_enabled - Reclaimable space enabled indicator |
| AUTO_STORAGE_HYBRID | SMALLINT | auto_storage_hybrid - Hybrid automatic storage table space indicator |
| TBSP_PATHS_DROPPED | SMALLINT | tablespace_paths_dropped - Table space using dropped path |
| POOL_DATA_GBP_L_READS | BIGINT | pool_data_gbp_l_reads - Group buffer pool data logical reads monitor element |
| POOL_DATA_GBP_P_READS | BIGINT | pool_data_gbp_p_reads - Group buffer pool data physical reads monitor element |
| POOL_DATA_LBP_PAGES_FOUND | BIGINT | pool_data_lbp_pages_found - Local buffer pool found data pages monitor element |
| POOL_DATA_GBP_INVALID_PAGES | BIGINT | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages monitor element |
| POOL_INDEX_GBP_L_READS | BIGINT | pool_index_gbp_l_reads - Group buffer pool index logical reads monitor element |
| POOL_INDEX_GBP_P_READS | BIGINT | pool_index_gbp_p_reads - Group buffer pool index physical reads monitor elements |
| POOL_INDEX_LBP_PAGES_FOUND | BIGINT | pool_index_lbp_pages_found - Local buffer pool index pages found monitor element |
| POOL_INDEX_GBP_INVALID_PAGES | BIGINT | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages monitor element |
| POOL_ASYNC_DATA_GBP_L_READS | BIGINT | pool_async_data_gbp_l_reads - Group buffer pool asynchronous data logical reads monitor element |
| POOL_ASYNC_DATA_GBP_P_READS | BIGINT | pool_async_data_gbp_p_reads - Group buffer pool asynchronous data physical reads monitor element |
| POOL_ASYNC_DATA_LBP_PAGES_FOUND | BIGINT | pool_async_data_lbp_pages_found - Local buffer pool asynchronous data pages found monitor element |
| POOL_ASYNC_DATA_GBP_INVALID_PAGES | BIGINT | pool_async_data_gbp_invalid_pages - Group buffer pool invalid asynchronous data pages monitor element |

Table 163. Information returned for MON_GET_TABLESPACE (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|------------------------------------|-----------|--|
| POOL_ASYNC_INDEX_GBP_L_READS | BIGINT | pool_async_index_gbp_l_reads - Group buffer pool asynchronous index logical reads monitor element |
| POOL_ASYNC_INDEX_GBP_P_READS | BIGINT | pool_async_index_gbp_p_reads - Group buffer pool asynchronous index physical reads monitor element |
| POOL_ASYNC_INDEX_LBP_PAGES_FOUND | BIGINT | pool_async_index_lbp_pages_found - Local buffer pool asynchronous index pages found monitor element |
| POOL_ASYNC_INDEX_GBP_INVALID_PAGES | BIGINT | pool_async_index_gbp_invalid_pages - Group buffer pool invalid asynchronous index pages monitor element |
| TABLESPACE_MIN_RECOVERY_TIME | TIMESTAMP | tablespace_min_recovery_time - Minimum recovery time for rollforward monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element

For a partitioned database environment, this will be the same value as for the MEMBER column. For Db2 Enterprise Server Edition and in a Db2 pureScale environment, this value will be 0.
Note: DBPARTITIONNUM is different to data_partition_id , which is used to identify a data partition that was created by subdividing data in a table based on a value. |
| POOL_XDA_GBP_L_READS | BIGINT | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |
| POOL_XDA_GBP_P_READS | BIGINT | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |
| POOL_XDA_LBP_PAGES_FOUND | BIGINT | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |
| POOL_XDA_GBP_INVALID_PAGES | BIGINT | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| POOL_ASYNC_XDA_GBP_L_READS | BIGINT | pool_async_xda_gbp_l_reads - Group buffer pool XDA data asynchronous logical read requests |
| POOL_ASYNC_XDA_GBP_P_READS | BIGINT | pool_async_xda_gbp_p_reads - Group buffer pool XDA data asynchronous physical read requests |
| POOL_ASYNC_XDA_LBP_PAGES_FOUND | BIGINT | pool_async_xda_lbp_pages_found - Asynchronous local buffer pool XDA data pages found |
| POOL_ASYNC_XDA_GBP_INVALID_PAGES | BIGINT | pool_async_xda_gbp_invalid_pages - Asynchronous group buffer pool invalid XDA data pages |
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer Pool Asynchronous Read Time monitor element |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time monitor element |

Table 163. Information returned for MON_GET_TABLESPACE (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|------------------------------------|--------------|--|
| TBSP_TRACKMOD_STATE | VARCHAR(32) | tbbsp_trackmod_state - Table space trackmod state.. This interface returns a text identifier, and is one of: <ul style="list-style-type: none"> • CLEAN • DIRTY • ININCREMENTAL • READFULL • READINCREMENTAL • UNAVAILABLE |
| STORAGE_GROUP_NAME | VARCHAR(128) | storage_group_name - Storage group name monitor element |
| STORAGE_GROUP_ID | INTEGER | storage_group_id - Storage group identifier monitor element |
| TBSP_DATATAG | SMALLINT | tbbsp_datatag - Table space data tag monitor element |
| TBSP_LAST_CONSEC_PAGE | BIGINT | tbbsp_last_consec_page - Last consecutive object table page monitor element |
| POOL_QUEUED_ASYNC_DATA_REQS | BIGINT | pool_queued_async_data_reqs - Data prefetch requests monitor element |
| POOL_QUEUED_ASYNC_INDEX_REQS | BIGINT | pool_queued_async_index_reqs - Index prefetch requests monitor element |
| POOL_QUEUED_ASYNC_XDA_REQS | BIGINT | pool_queued_async_xda_reqs - XDA prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_OTHER_REQS | BIGINT | pool_queued_async_other_reqs - Non-prefetch requests monitor element |
| POOL_QUEUED_ASYNC_DATA_PAGES | BIGINT | pool_queued_async_data_pages - Data pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_INDEX_PAGES | BIGINT | pool_queued_async_index_pages - Index pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_XDA_PAGES | BIGINT | pool_queued_async_xda_pages - XDA pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES | BIGINT | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES | BIGINT | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_DATA_REQS | BIGINT | pool_failed_async_data_reqs - Failed data prefetch requests monitor element |

Table 163. Information returned for MON_GET_TABLESPACE (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|---|
| POOL_FAILED_ASYNC_INDEX_REQS | BIGINT | pool_failed_async_index_reqs - Failed index prefetch requests monitor element |
| POOL_FAILED_ASYNC_XDA_REQS | BIGINT | pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element |
| POOL_FAILED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_OTHER_REQS | BIGINT | pool_failed_async_other_reqs - Failed non-prefetch requests monitor element |
| SKIPPED_PREFETCH_DATA_P_READS | BIGINT | skipped_prefetch_data_p_reads - Skipped prefetch data physical reads monitor element |
| SKIPPED_PREFETCH_INDEX_P_READS | BIGINT | skipped_prefetch_index_p_reads - Skipped prefetch index physical reads monitor element |
| SKIPPED_PREFETCH_XDA_P_READS | BIGINT | skipped_prefetch_xda_p_reads - Skipped prefetch XDA physical reads monitor element |
| SKIPPED_PREFETCH_TEMP_DATA_P_READS | BIGINT | skipped_prefetch_temp_data_p_reads - Skipped prefetch temporary data physical reads monitor element |
| SKIPPED_PREFETCH_TEMP_INDEX_P_READS | BIGINT | skipped_prefetch_temp_index_p_reads - Skipped prefetch temporary index physical reads monitor element |
| SKIPPED_PREFETCH_TEMP_XDA_P_READS | BIGINT | skipped_prefetch_temp_xda_p_reads - Skipped prefetch temporary XDA data physical reads monitor element |
| SKIPPED_PREFETCH_UOW_DATA_P_READS | BIGINT | skipped_prefetch_uow_data_p_reads - Skipped prefetch unit of work data physical reads monitor element |
| SKIPPED_PREFETCH_UOW_INDEX_P_READS | BIGINT | skipped_prefetch_uow_index_p_reads - Skipped prefetch unit of work index physical reads monitor element |
| SKIPPED_PREFETCH_UOW_XDA_P_READS | BIGINT | skipped_prefetch_uow_xda_p_reads - Skipped prefetch unit of work XDA data physical reads monitor element |
| SKIPPED_PREFETCH_UOW_TEMP_DATA_P_READS | BIGINT | skipped_prefetch_uow_temp_data_p_reads - Skipped prefetch unit of work temporary data physical reads monitor element |
| SKIPPED_PREFETCH_UOW_TEMP_INDEX_P_READS | BIGINT | skipped_prefetch_uow_temp_index_p_reads - Skipped prefetch unit of work temporary index physical reads monitor element |
| SKIPPED_PREFETCH_UOW_TEMP_XDA_P_READS | BIGINT | skipped_prefetch_uow_temp_xda_p_reads - Skipped prefetch unit of work temporary XDA data physical reads monitor element |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |

Table 163. Information returned for MON_GET_TABLESPACE (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|---|
| PREFETCH_WAITS | BIGINT | prefetch_waits - Prefetcher wait count monitor element |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |
| POOL_ASYNC_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_async_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found by asynchronous EDUs in a local buffer pool monitor element monitor element |
| POOL_ASYNC_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_async_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found by asynchronous EDUs in a local buffer pool monitor element monitor element |
| POOL_ASYNC_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_async_xda_gbp_indep_pages_found_in_lbp - Group buffer pool independent XML storage object(XDA) pages found by asynchronous EDUs in a local buffer pool monitor element monitor element |
| TBSP_NUM_CONTAINERS | BIGINT | tablespace_num_containers - Number of Containers in Table Space |
| TBSP_INITIAL_SIZE | BIGINT | tablespace_initial_size - Initial table space size |
| TBSP_MAX_SIZE | BIGINT | tablespace_max_size - Maximum table space size |
| TBSP_INCREASE_SIZE | BIGINT | tablespace_increase_size - Increase size in bytes |
| TBSP_INCREASE_SIZE_PERCENT | SMALLINT | tablespace_increase_size_percent - Increase size by percent |
| TBSP_LAST_RESIZE_TIME | TIMESTAMP | tablespace_last_resize_time - Time of last successful resize |
| TBSP_LAST_RESIZE_FAILED | SMALLINT | tablespace_last_resize_failed - Last resize attempt failed |
| POOL_COL_L_READS | BIGINT | pool_col_l_reads - Buffer pool column-organized logical reads monitor element |
| POOL_TEMP_COL_L_READS | BIGINT | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| POOL_COL_P_READS | BIGINT | pool_col_p_reads - Buffer pool column-organized physical reads |
| POOL_TEMP_COL_P_READS | BIGINT | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads monitor element |
| POOL_COL_LBP_PAGES_FOUND | BIGINT | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found monitor element |
| POOL_COL_WRITES | BIGINT | pool_col_writes - Buffer pool column-organized writes |

Table 163. Information returned for MON_GET_TABLESPACE (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|---|
| POOL_ASYNC_COL_READS | BIGINT | pool_async_col_reads - Buffer pool asynchronous column-organized reads |
| POOL_ASYNC_COL_READ_REQS | BIGINT | pool_async_col_read_reqs - Buffer pool asynchronous column-organized read requests |
| POOL_ASYNC_COL_WRITES | BIGINT | pool_async_col_writes - Buffer pool asynchronous column-organized writes |
| POOL_ASYNC_COL_LBP_PAGES_FOUND | BIGINT | pool_async_col_lbp_pages_found - Asynchronous local buffer pool column-organized pages found |
| POOL_COL_GBP_L_READS | BIGINT | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| POOL_COL_GBP_P_READS | BIGINT | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| POOL_COL_GBP_INVALID_PAGES | BIGINT | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |
| POOL_ASYNC_COL_GBP_L_READS | BIGINT | pool_async_col_gbp_l_reads - Asynchronous group buffer pool column-organized logical reads |
| POOL_ASYNC_COL_GBP_P_READS | BIGINT | pool_async_col_gbp_p_reads - Asynchronous group buffer pool column-organized physical reads |
| POOL_ASYNC_COL_GBP_INVALID_PAGES | BIGINT | pool_async_col_gbp_invalid_pages - Asynchronous group buffer pool invalid data pages monitor element |
| POOL_ASYNC_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_async_col_gbp_indep_pages_found_in_lbp - Asynchronous group buffer pool column-organized independent data pages found in local buffer pool monitor element |
| POOL_QUEUED_ASYNC_COL_REQS | BIGINT | pool_queued_async_col_reqs - Column-organized prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_REQS | BIGINT | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| POOL_QUEUED_ASYNC_COL_PAGES | BIGINT | pool_queued_async_col_pages - Column-organized page prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_PAGES | BIGINT | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| POOL_FAILED_ASYNC_COL_REQS | BIGINT | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| POOL_FAILED_ASYNC_TEMP_COL_REQS | BIGINT | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |
| SKIPPED_PREFETCH_COL_P_READS | BIGINT | skipped_prefetch_col_p_reads - Skipped prefetch column-organized physical reads |
| SKIPPED_PREFETCH_TEMP_COL_P_READS | BIGINT | skipped_prefetch_temp_col_p_reads - Skipped prefetch column-organized temporary physical reads |
| SKIPPED_PREFETCH_UOW_COL_P_READS | BIGINT | skipped_prefetch_uow_col_p_reads - Skipped prefetch unit of work column-organized physical reads |

Table 163. Information returned for MON_GET_TABLESPACE (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|------------|---|
| SKIPPED_PREFETCH_UOW_TEMP_COL_P_READS | BIGINT | skipped_prefetch_uow_temp_col_p_reads - Skipped prefetch unit of work column-organized temporary physical reads |
| TBSP_NUM QUIESCERS | BIGINT | tablespace_num_quiescers |
| TBSP_NUM_RANGES | BIGINT | tablespace_num_ranges |
| CACHING_TIER | VARCHAR(8) | Reserved for future use. |
| POOL_DATA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_READ_TIME | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_WRITE_TIME | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |

Table 163. Information returned for MON_GET_TABLESPACE (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|--|-----------|--|
| POOL_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_READS | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_READS | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_READS | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_READS | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |

Table 163. Information returned for MON_GET_TABLESPACE (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|--|
| POOL_ASYNC_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_ASYNC_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_ASYNC_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| LOB_PREFETCH_WAIT_TIME | BIGINT | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| LOB_PREFETCH_REQS | BIGINT | lob_prefetch_reqs - LOB prefetcher request count monitor element |

MON_GET_TABLESPACE QUIESCER table function - Get information about quiesced table spaces

The MON_GET_TABLESPACE QUIESCER table function returns information about quiesced table spaces. A single row is returned for each member.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- SQLADM authority
- DBADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_GET_TABLESPACE QUIESCER—(—*member*—)—————►►

The schema is SYSPROC.

Routine parameters

member

An optional input argument of type SMALLINT that specifies a valid member in the same instance as the currently connected database when you call this function. Specify -1 for the current database member or -2 for all active database members. If the null value is specified, -1 is set.

Information returned

Table 164. Information returned by MON_GET_TABLESPACE QUIESCER

| Column name | Data type | Description |
|-----------------------------|--------------|---|
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name monitor element |
| QUIESCER_TS_ID | BIGINT | quiescer_ts_id - Quiescer table space identification monitor element |
| QUIESCER_OBJ_ID | BIGINT | quiescer_obj_id - Quiescer object identification monitor element |
| QUIESCER_AUTH_ID | VARCHAR(128) | quiescer_auth_id - Quiescer user authorization identification monitor element |
| QUIESCER_APPLICATION_HANDLE | BIGINT | quiescer_application_handle - Quiescer application handle monitor element |
| QUIESCER_STATE | VARCHAR(14) | quiescer_state - Quiescer state monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

Usage notes

Use the MON_GET_TABLESPACE QUIESCER table function in place of the SNAP_GET_TBSP QUIESCER table function and SNAPTbsp QUIESCER view, which have been deprecated.

Examples

- Retrieve information about the quiesced table spaces for all database members for the currently connected database.

```
SELECT SUBSTR(TBSP_NAME, 1, 10) AS TBSP_NAME, QUIESCER_TS_ID,
       QUIESCER_OBJ_ID, QUIESCER_AUTH_ID, QUIESCER_APPLICATION_HANDLE,
       QUIESCER_STATE, DBPARTITIONNUM
FROM TABLE(MON_GET_TABLESPACE QUIESCER(-2)) ORDER BY DBPARTITIONNUM
```

Sample output from this query is as follows:

```
TBSP_NAME QUIESCER_TS_ID QUIESCER_OBJ_ID QUIESCER_AUTH_ID ...
-----
USERSPACE1          2          5 SWALKTY ...
USERSPACE1          2          5 SWALKTY ...
```

2 record(s) selected.

Output from this query (continued):

```
... QUIESCER_APPLICATION_HANDLE QUIESCER_STATE DBPARTITIONNUM
... -----
...          0 EXCLUSIVE          0
...        65983 EXCLUSIVE          1
```

- Retrieve information about the quiesced table spaces for database member 1 for the currently connected database.

```
SELECT SUBSTR(TBSP_NAME, 1, 10) AS TBSP_NAME, QUIESCER_TS_ID,
       QUIESCER_OBJ_ID, QUIESCER_AUTH_ID, QUIESCER_APPLICATION_HANDLE,
       QUIESCER_STATE, DBPARTITIONNUM
FROM TABLE( MON_GET_TABLESPACE QUIESCER(1)) AS T
```

The following is sample output from this query:

| TBSP_NAME | QUIESCER_TS_ID | QUIESCER_OBJ_ID | QUIESCER_AUTH_ID | ... |
|------------|----------------|-----------------|------------------|-----|
| USERSPACE1 | 2 | 5 | SWALKTY | ... |

1 record(s) selected.

Output from this query (continued):

| ... | QUIESCER_APPLICATION_HANDLE | QUIESCER_STATE | DBPARTITIONNUM |
|-----|-----------------------------|----------------|----------------|
| ... | 65983 | EXCLUSIVE | 1 |

- Determine the table name for a quiesced table in a partitioned database.

If the table is partitioned and kept in quiesced state, the values for table space ID and table ID are different from the values that are stored in SYSCAT.TABLES. The values for these IDs are stored with the unsigned short representation. The first step in finding the table name of the quiesced table is to find the signed short representation. Do this by calculating the table space ID by subtracting 65536 (the maximum value) from QUIESCER_TS_ID. Next, use this table space ID to locate the quiesced tables. The actual table space ID can be found in SYSCAT.DATAPARTITIONS for each data partition in the table.

```
SELECT SUBSTR(TBSP_NAME, 1, 10) AS TBSP_NAME,
       CASE WHEN QUIESCER_TS_ID = 65530
            THEN QUIESCER_TS_ID - 65536
            ELSE QUIESCER_TS_ID END as tbspaceid,
       CASE WHEN QUIESCER_TS_ID = 65530
            THEN QUIESCER_OBJ_ID - 65536
            ELSE QUIESCER_OBJ_ID END as tableid
FROM TABLE(MON_GET_TABLESPACE QUIESCER(-2))
ORDER BY DBPARTITIONNUM
```

Sample output from this query is as follows:

| TBSP_NAME | TBSPACEID | TABLEID |
|-----------|-----------|---------|
| TABDATA | -6 | -32768 |
| DATAMART | -6 | -32765 |
| SMALL | 5 | 17 |

3 record(s) selected.

Use the given TBSPACEID and TABLEID provided in the output from the previous query to find the table schema and name from SYSCAT.TABLES:

- For the table spaces TABDATA and DATAMART, run the following query:

```
SELECT CHAR(tabschema, 10) tabschema, CHAR(tabname, 15) tabname
FROM SYSCAT.TABLES
WHERE tbspaceid = -6 AND tableid in (-32768, -32765)
```

Sample output from this query is as follows:

| TABSCHEMA | TABNAME |
|-----------|--------------|
| SAMP | ORDERS_RP |
| SAMP | ORDERS_DMART |

2 record(s) selected.

- For the table space SMALL, run the following query:

```
SELECT CHAR(tabschema, 10)tabschema, CHAR(tabname,15)tabname
FROM SYSCAT.TABLES
WHERE tbspaceid = 5 AND tableid = 17
```

Sample output from this query is as follows:

| TABSCHEMA | TABNAME |
|-----------|---------|
| SAMP | NATION |

1 record(s) selected.

MON_GET_TABLESPACE_RANGE table function - Get information about table space ranges

The MON_GET_TABLESPACE_RANGE table function returns information about table space ranges. A single row is returned for each member.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- SQLADM authority
- DBADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_GET_TABLESPACE_RANGE—(—*dbpartitionnum*—)—————►◄

The schema is SYSPROC.

Routine parameters

dbpartitionnum

An optional input argument of type INTEGER that specifies a valid database partition number. Specify *-1* for the current database partition, or *-2* for an aggregate of all active database partitions. An active database partition is a partition where the database is available for connection and use by applications. If the null value is specified, *-1* is set implicitly.

Information returned

Table 165. Information returned by MON_GET_TABLESPACE_RANGE

| Column name | Data type | Description |
|-------------------------|--------------|---|
| TBSP_ID | BIGINT | tablespace_id - Table space identification monitor element |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name monitor element |
| RANGE_NUMBER | BIGINT | range_number - Range Number monitor element |
| RANGE_STRIPE_SET_NUMBER | BIGINT | range_stripe_set_number - Stripe set number monitor element |
| RANGE_OFFSET | BIGINT | range_offset - Range offset monitor element |
| RANGE_MAX_PAGE | BIGINT | range_max_page_number - Maximum page in range monitor element |
| RANGE_MAX_EXTENT | BIGINT | range_max_extent - Maximum extent in range monitor element |
| RANGE_START_STRIPE | BIGINT | range_start_stripe - Start stripe monitor element |
| RANGE_END_STRIPE | BIGINT | range_end_stripe - End stripe monitor element |
| RANGE_ADJUSTMENT | BIGINT | range_adjustment - Range adjustment monitor element |
| RANGE_NUM_CONTAINER | BIGINT | range_num_container - Number of containers in range monitor element |
| RANGE_CONTAINER_ID | BIGINT | range_container_id - Range container monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

Usage notes

Use the MON_GET_TABLESPACE_RANGE table function in place of the SNAP_GET_TBSP_RANGE table function and SNAPTbsp_RANGE view, which have been deprecated.

Examples

- Select information about table space ranges for all database partitions for the currently connected database.

```
SELECT TBSP_ID, SUBSTR(TBSP_NAME, 1, 15) AS TBSP_NAME, RANGE_NUMBER,  
       RANGE_STRIPE_SET_NUMBER, RANGE_OFFSET, RANGE_MAX_PAGE,  
       RANGE_MAX_EXTENT, RANGE_START_STRIPE, RANGE_END_STRIPE,  
       RANGE_ADJUSTMENT, RANGE_NUM_CONTAINER, RANGE_CONTAINER_ID,  
       DBPARTITIONNUM FROM TABLE(MON_GET_TABLESPACE_RANGE(-2))  
ORDER BY DBPARTITIONNUM
```

The following is sample output from this query:

| TBSP_ID | TBSP_NAME | RANGE_NUMBER | RANGE_STRIPE_SET_NUMBER | ... |
|---------|--------------|--------------|-------------------------|-----|
| 0 | SYSCATSPACE | 0 | 0 | ... |
| 2 | USERSPACE1 | 0 | 0 | ... |
| 3 | SYSTOOLSPACE | 0 | 0 | ... |
| 2 | USERSPACE1 | 0 | 0 | ... |
| 2 | USERSPACE1 | 0 | 0 | ... |

5 record(s) selected.

Output from this query (continued):

| ... | RANGE_OFFSET | RANGE_MAX_PAGE | RANGE_MAX_EXTENT | ... |
|-----|--------------|----------------|------------------|-----|
| ... | 0 | 11515 | 2878 | ... |
| ... | 0 | 479 | 14 | ... |
| ... | 0 | 251 | 62 | ... |
| ... | 0 | 479 | 14 | ... |
| ... | 0 | 479 | 14 | ... |

Output from this query (continued):

| ... | RANGE_START_STRIPE | RANGE_END_STRIPE | RANGE_ADJUSTMENT | ... |
|-----|--------------------|------------------|------------------|-----|
| ... | 0 | 2878 | 0 | ... |
| ... | 0 | 14 | 0 | ... |
| ... | 0 | 62 | 0 | ... |
| ... | 0 | 14 | 0 | ... |
| ... | 0 | 14 | 0 | ... |

Output from this query (continued):

| ... | RANGE_NUM_CONTAINER | RANGE_CONTAINER_ID | DBPARTITIONNUM |
|-----|---------------------|--------------------|----------------|
| ... | 1 | 0 | 0 |
| ... | 1 | 0 | 0 |
| ... | 1 | 0 | 0 |
| ... | 1 | 0 | 1 |
| ... | 1 | 0 | 2 |

- Select information about the table space range for the table space whose TBSP_ID value is 2 on the current database partition.

```
SELECT TBSP_ID, SUBSTR(TBSP_NAME, 1, 15) AS TBSP_NAME, RANGE_NUMBER,
       RANGE_STRIPE_SET_NUMBER, RANGE_OFFSET, RANGE_MAX_PAGE, RANGE_MAX_EXTENT,
       RANGE_START_STRIPE, RANGE_END_STRIPE, RANGE_ADJUSTMENT,
       RANGE_NUM_CONTAINER, RANGE_CONTAINER_ID
FROM TABLE(MON_GET_TABLESPACE_RANGE(-1)) AS T WHERE TBSP_ID = 2
```

The following is sample output from this query:

| TBSP_ID | TBSP_NAME | RANGE_NUMBER | ... |
|---------|------------|--------------|-----|
| 2 | USERSPACE1 | 0 | ... |

1 record(s) selected.

Output from this query (continued):

| ... | RANGE_STRIPE_SET_NUMBER | RANGE_OFFSET | RANGE_MAX_PAGE | ... |
|-----|-------------------------|--------------|----------------|-----|
| ... | 0 | 0 | 3967 | ... |

Output from this query (continued):

| ... | RANGE_MAX_EXTENT | RANGE_START_STRIPE | RANGE_END_STRIPE | ... |
|-----|------------------|--------------------|------------------|-----|
| ... | 123 | 0 | 123 | ... |

Output from this query (continued):

| ... RANGE_ADJUSTMENT | RANGE_NUM_CONTAINER | RANGE_CONTAINER_ID |
|----------------------|---------------------|--------------------|
| ... | 0 | 1 |
| ... | | 0 |

MON_GET_TABLE_USAGE_LIST table function - Returns information from a table usage list

The MON_GET_TABLE_USAGE_LIST table function returns information from a usage list defined for a table.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_GET_TABLE_USAGE_LIST—(*—usagelistschema—*,*—usagelistname—*,*—member—*)—►◄

The schema is SYSPROC.

Routine parameters

usagelistschema

An input argument of type VARCHAR(128) that specifies a valid schema name in the currently connected database when calling this function. If the argument is null or an empty string, usage lists are retrieved in all schemas in the database. If the argument is specified, usage lists are only returned for the specified schema.

usagelistname

An input argument of type VARCHAR(128) that specifies a usage list defined for a table that resides in the currently connected database when calling this function. If *usagelistname* is null or an empty string, then all usage lists defined for a table from the schemas identified by the *usagelistschema* that exist are retrieved. If specified, only the usage list specified from the schemas identified by the *usagelistschema* is returned.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is set implicitly.

Information returned

Table 166. Information returned for MON_GET_TABLE_USAGE_LIST

| Column Name | Data Type | Description |
|-------------------------|--------------------------|---|
| USAGELISTSHEMA | VARCHAR (128) | usage_list_schema - Usage list schema |
| USAGELISTNAME | VARCHAR (128) | usage_list_name - Usage list name |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TABNAME | VARCHAR(128) | table_name - Table name |
| MEMBER | SMALLINT | member - Database member |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data partition identifier |
| EXECUTABLE_ID | VARCHAR(32) FOR BIT DATA | executable_id - Executable ID |
| MON_INTERVAL_ID | BIGINT | mon_interval_id - Monitor interval identifier |
| LAST_UPDATED | TIMESTAMP | last_updated - Last update time stamp |
| NUM_REFERENCES | BIGINT | num_references - Number of references |
| NUM_REF_WITH_METRICS | BIGINT | num_ref_with_metrics - Number of references with metrics |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated |
| ROWS_READ | BIGINT | rows_read - Rows read |
| OVERFLOW_CREATES | BIGINT | overflow_creates - Overflow creates |
| OVERFLOW_ACCESSES | BIGINT | overflow_accesses - Accesses to overflowed records |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_WAIT_TIME_GLOBAL | BIGINT | lock_wait_time_global - Lock wait time global |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOCK_WAITS_GLOBAL | BIGINT | lock_waits_global - Lock waits global |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| LOCK_ESCALS_GLOBAL | BIGINT | lock_escals_global - Number of global lock escalations |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| OBJECT_DATA_L_READS | BIGINT | object_data_l_reads - Buffer pool data logical reads for a table |
| OBJECT_DATA_P_READS | BIGINT | object_data_p_reads - Buffer pool physical data reads for a table |
| OBJECT_DATA_GBP_L_READS | BIGINT | object_data_gbp_l_reads - GBP data logical reads for a table |
| OBJECT_DATA_GBP_P_READS | BIGINT | object_data_gbp_p_reads - GBP data physical reads for a table |

Table 166. Information returned for MON_GET_TABLE_USAGE_LIST (continued)

| Column Name | Data Type | Description |
|--|-----------|---|
| OBJECT_DATA_GBP_INVALID_PAGES | BIGINT | object_data_gbp_invalid_pages - GBP invalid data pages for a table |
| OBJECT_DATA_LBP_PAGES_FOUND | BIGINT | object_data_lbp_pages_found - LBP data pages found for a table |
| OBJECT_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | object_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool |
| OBJECT_XDA_L_READS | BIGINT | object_xda_l_reads - Buffer pool XDA data logical reads for a table |
| OBJECT_XDA_P_READS | BIGINT | object_xda_p_reads - Buffer pool XDA data physical reads for a table |
| OBJECT_XDA_GBP_L_READS | BIGINT | object_xda_gbp_l_reads - GBP XDA data logical read requests for a table |
| OBJECT_XDA_GBP_P_READS | BIGINT | object_xda_gbp_p_reads - GBP XDA data physical read requests for a table |
| OBJECT_XDA_GBP_INVALID_PAGES | BIGINT | object_xda_gbp_invalid_pages - GBP invalid XDA data pages for a table |
| OBJECT_XDA_LBP_PAGES_FOUND | BIGINT | object_xda_lbp_pages_found - LBP XDA data pages found for a table |
| OBJECT_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | object_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool |
| OBJECT_COL_L_READS | BIGINT | object_col_l_reads - Column-organized logical reads |
| OBJECT_COL_P_READS | BIGINT | object_col_p_reads - Column-organized physical reads |
| OBJECT_COL_GBP_L_READS | BIGINT | object_col_gbp_l_reads - Group buffer pool column-organized logical reads |
| OBJECT_COL_GBP_P_READS | BIGINT | object_col_gbp_p_reads - Group buffer pool column-organized physical reads |
| OBJECT_COL_GBP_INVALID_PAGES | BIGINT | object_col_gbp_invalid_pages - Group buffer pool column-organized invalid data pages |
| OBJECT_COL_LBP_PAGES_FOUND | BIGINT | object_col_lbp_pages_found - Local buffer pool column-organized pages found |
| OBJECT_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | object_col_gbp_indep_pages_found_in_lbp - Group buffer pool column-organized index pages found in local buffer pool |
| OBJECT_DATA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| OBJECT_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| OBJECT_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| OBJECT_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| OBJECT_XDA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |

Table 166. Information returned for `MON_GET_TABLE_USAGE_LIST` (continued)

| Column Name | Data Type | Description |
|---|-----------|--|
| OBJECT_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| OBJECT_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| OBJECT_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| OBJECT_COL_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| OBJECT_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| OBJECT_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| OBJECT_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| EXT_TABLE_RECV_WAIT_TIME | BIGINT | ext_table_recv_wait_time - Total agent wait time for external table readers monitor element |
| EXT_TABLE_RECVS_TOTAL | BIGINT | ext_table_recvs_total - Total row batches received from external table readers monitor element |
| EXT_TABLE_RECV_VOLUME | BIGINT | ext_table_recv_volume - Total data received from external table readers monitor element |
| EXT_TABLE_READ_VOLUME | BIGINT | ext_table_read_volume - Total data read by external table readers monitor element |
| EXT_TABLE_SEND_WAIT_TIME | BIGINT | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| EXT_TABLE_SENDS_TOTAL | BIGINT | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| EXT_TABLE_SEND_VOLUME | BIGINT | ext_table_send_volume - Total data sent to external table writers monitor element |
| EXT_TABLE_WRITE_VOLUME | BIGINT | ext_table_write_volume - Total data written by external table writers monitor element |

Usage notes

Each row returned by this function represents the total number of times (num_references) a unique section (DML statement only, executable ID) has referenced a particular object during a particular time interval (monitor interval ID) since being added to the list. The statistics collected for this row represents the total aggregated value across these executions during this time interval.

Use the num_ref_with_metrics column instead of the num_references column when computing averages, since the num_references column counts all executions of the section, regardless of whether or not the execution of the section contributed to the metrics that are reported.

Metrics collected by this function are controlled at the database level using the `mon_obj_metrics` configuration parameter. By default, metrics collection is enabled.

Example

Retrieve the usage list USL_MON_PAYROLL from member 3

```
SELECT * FROM TABLE(
    MON_GET_TABLE_USAGE_LIST(NULL, 'USL_MON_PAYROLL', 3))
```

| USAGELISTSHEMA | USAGELISTNAME | TABSCHEMA | TABNAME |
|----------------|-----------------|-----------|---------|
| ISAYYID | USL_MON_PAYROLL | ISAYYID | T1 |
| ISAYYID | USL_MON_PAYROLL | ISAYYID | T1 |

| MEMBER | LAST_UPDATED |
|--------|----------------------------|
| 3 | 2011-07-06-10.20.22.727803 |
| 3 | 2011-07-06-10.20.58.202161 |

2 record(s) selected.

MON_GET_TRANSACTION_LOG table function - Get log information

The `MON_GET_TRANSACTION_LOG` table function returns information about the transaction logging subsystem for the currently connected database.

Syntax

```
►►—MON_GET_TRANSACTION_LOG—(—member—)—————►►
```

The schema is SYSPROC.

Table function parameters

member

An input argument of type `INTEGER` that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Example

```
Select MEMBER, CUR_COMMIT_DISK_LOG_READS, CURRENT_ACTIVE_LOG,
APPLID_HOLDING_OLDEST_XACT from table(mon_get_transaction_log(-1)) as t
order by member asc
```

| | | | |
|--------|---------------------------|--------------------|----------------------------|
| MEMBER | CUR_COMMIT_DISK_LOG_READS | CURRENT_ACTIVE_LOG | APPLID_HOLDING_OLDEST_XACT |
| ----- | ----- | ----- | ----- |
| 0 | 9999 | 1 | 7 |

Information returned

Table 167. Information returned for MON_GET_TRANSACTION_LOG. There will be one row per member for both Db2 pureScale and partitioned database environments.

| Column Name | Data Type | Description |
|------------------------------|-----------|---|
| MEMBER | SMALLINT | member - Database member monitor element |
| TOTAL_LOG_AVAILABLE | BIGINT | total_log_available - Total Log Available monitor element |
| TOTAL_LOG_USED | BIGINT | total_log_used - Total Log Space Used monitor element |
| SEC_LOG_USED_TOP | BIGINT | sec_log_used_top - Maximum Secondary Log Space Used monitor element |
| TOT_LOG_USED_TOP | BIGINT | tot_log_used_top - Maximum Total Log Space Used monitor element |
| SEC_LOGS_ALLOCATED | BIGINT | sec_logs_allocated - Secondary Logs Allocated Currently monitor element |
| LOG_READS | BIGINT | log_reads - Number of Log Pages Read monitor element |
| LOG_READ_TIME | BIGINT | log_read_time - Log Read Time monitor element |
| LOG_WRITES | BIGINT | log_writes - Number of Log Pages Written monitor element |
| LOG_WRITE_TIME | BIGINT | log_write_time - Log Write Time monitor element |
| NUM_LOG_WRITE_IO | BIGINT | num_log_write_io - Number of Log Writes monitor element |
| NUM_LOG_READ_IO | BIGINT | num_log_read_io - Number of Log Reads monitor element |
| NUM_LOG_PART_PAGE_IO | BIGINT | num_log_part_page_io - Number of Partial Log Page Writes monitor element |
| NUM_LOG_BUFFER_FULL | BIGINT | num_log_buffer_full - Number of times full log buffer caused agents to wait monitor element |
| NUM_LOG_DATA_FOUND_IN_BUFFER | BIGINT | num_log_data_found_in_buffer - Number of Log Data Found In Buffer monitor element |
| APPLID_HOLDING_OLDEST_XACT | BIGINT | Application handle holding the oldest transaction |
| LOG_TO_REDO_FOR_RECOVERY | BIGINT | log_to_redo_for_recovery - Amount of Log to be Redone for Recovery monitor element |
| LOG_HELD_BY_DIRTY_PAGES | BIGINT | log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages monitor element |
| FIRST_ACTIVE_LOG | BIGINT | first_active_log - First Active Log File Number monitor element |
| LAST_ACTIVE_LOG | BIGINT | last_active_log - Last Active Log File Number monitor element |
| CURRENT_ACTIVE_LOG | BIGINT | current_active_log - Current® Active Log File Number monitor element |
| CURRENT_ARCHIVE_LOG | BIGINT | current_archive_log - Current Archive Log File Number monitor element |
| CUR_COMMIT_DISK_LOG_READS | BIGINT | The number of times the currently committed version of a row was retrieved via a log read from disk (versus log buffer) |

Table 167. Information returned for `MON_GET_TRANSACTION_LOG` (continued). There will be one row per member for both Db2 pureScale and partitioned database environments.

| Column Name | Data Type | Description |
|--|-----------|---|
| <code>CUR_COMMIT_TOTAL_LOG_READS</code> | BIGINT | The total number of times the currently committed version of a row was retrieved from the logs (log buffer and disk) |
| <code>CUR_COMMIT_LOG_BUFF_LOG_READS</code> | BIGINT | This is calculated as <code>CUR_COMMIT_TOTAL_LOG_READS - CUR_COMMIT_DISK_LOG_READS</code> |
| <code>ARCHIVE_METHOD1_STATUS</code> | SMALLINT | The result of the most recent log archive attempt. Possible values are 1 for Success or 0 for Failure or Null if not applicable. |
| <code>METHOD1_NEXT_LOG_TO_ARCHIVE</code> | BIGINT | The next log file to be archived. |
| <code>METHOD1_FIRST_FAILURE</code> | BIGINT | The first log file that was unsuccessfully archived. |
| <code>ARCHIVE_METHOD2_STATUS</code> | SMALLINT | The result of the most recent secondary log archive attempt. Possible values are 1 for Success or 0 for Failure or Null if not applicable. |
| <code>METHOD2_NEXT_LOG_TO_ARCHIVE</code> | BIGINT | The next secondary log file to be archived. |
| <code>METHOD2_FIRST_FAILURE</code> | BIGINT | The first secondary log file that was unsuccessfully archived. |
| <code>LOG_CHAIN_ID</code> | BIGINT | The identifier of the log chain number |
| <code>CURRENT_LSO</code> | BIGINT | The current log sequence offset |
| <code>CURRENT_LSN</code> | BIGINT | The current log sequence number |
| <code>OLDEST_TX_LSN</code> | BIGINT | The log sequence number associated with the oldest transaction running. |
| <code>NUM_LOGS_AVAIL_FOR_RENAME</code> | INTEGER | If using log archiving, this is the number of log files maintained in the log path for the purpose of reusing them. Otherwise its value is 0. |
| <code>NUM_INDOUBT_TRANS</code> | BIGINT | <code>num_indoubt_trans</code> - Number of Indoubt Transactions monitor element |
| <code>LOG_HADR_WAIT_TIME</code> | BIGINT | Time spent waiting on HADR processing. |
| <code>LOG_HADR_WAITS_TOTAL</code> | BIGINT | Total number of times the transaction log sub-system waited for HADR processing. |

MON_GET_UNIT_OF_WORK table function - Get unit of work metrics

The `MON_GET_UNIT_OF_WORK` table function returns metrics for one or more units of work.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

```
►►—MON_GET_UNIT_OF_WORK—(—————►  
►—application_handle—,—member—  
                          └,—system_appls—┘)—————►
```

The schema is SYSPROC.

Routine parameters

application_handle

An optional input argument of type BIGINT that specifies a valid application handle in the same database as the one currently connected to when calling this function. If the argument is null, metrics are retrieved for units of work running in all superclasses in the database.

member

An optional input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is set implicitly.

system_appls

An input parameter of data type SMALLINT that specifies whether information for system applications is returned. The following values are valid:

- 0 or NULL: System application information is not returned. NULL is the default if a value for *system_appls* is not provided.
- 1: User and system application information is returned.

Information returned

Table 168. Information returned for MON_GET_UNIT_OF_WORK

| Column Name | Data Type | Description or corresponding monitor element |
|-------------------------|--------------|---|
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name |
| SERVICE_SUBCLASS_NAME | VARCHAR(128) | service_subclass_name - Service subclass name |
| SERVICE_CLASS_ID | INTEGER | service_class_id - Service class ID |
| MEMBER | SMALLINT | member - Database member |
| COORD_MEMBER | SMALLINT | coord_member - Coordinator member |
| APPLICATION_HANDLE | BIGINT | application_handle - Application handle |
| APPLICATION_ID | VARCHAR(128) | appl_id - Application ID |
| WORKLOAD_NAME | VARCHAR(128) | workload_name - Workload name |
| WORKLOAD_OCCURRENCE_ID | INTEGER | workload_occurrence_id - Workload occurrence identifier. This ID does not uniquely identify the workload occurrence unless it is coupled with the coordinator member and the workload name. |
| UOW_ID | INTEGER | uow_id - Unit of work ID |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---------------------------|--------------|---|
| WORKLOAD_OCCURRENCE_STATE | VARCHAR(32) | workload_occurrence_state - Workload occurrence state |
| CLIENT_WRKSTNNAME | VARCHAR(255) | CURRENT CLIENT_WRKSTNNAME special register |
| CLIENT_ACCTNG | VARCHAR(255) | CURRENT CLIENT_ACCTNG special register |
| CLIENT_USERID | VARCHAR(255) | CURRENT CLIENT_USERID special register |
| CLIENT_APPLNAME | VARCHAR(255) | CURRENT CLIENT_APPLNAME special register |
| UOW_START_TIME | TIMESTAMP | uow_start_time - Unit of Work Start Timestamp |
| SESSION_AUTH_ID | VARCHAR(128) | session_auth_id - Session authorization ID |
| ACT_ABORTED_TOTAL | BIGINT | act_aborted_total - Total aborted activities |
| ACT_COMPLETED_TOTAL | BIGINT | act_completed_total - Total completed activities |
| ACT_REJECTED_TOTAL | BIGINT | act_rejected_total - Total rejected activities |
| AGENT_WAIT_TIME | BIGINT | agent_wait_time - Agent wait time |
| AGENT_WAITS_TOTAL | BIGINT | agent_waits_total - Total agent waits |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer pool XDA data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer pool XDA data physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer pool XDA data writes |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|-----------------------|-----------|---|
| CLIENT_IDLE_WAIT_TIME | BIGINT | client_idle_wait_time - Client idle wait time |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| FCM_RECV_VOLUME | BIGINT | fcm_recv_volume - FCM recv volume |
| FCM_RECVS_TOTAL | BIGINT | fcm_recv_total - FCM recvs total |
| FCM_SEND_VOLUME | BIGINT | fcm_send_volume - FCM send volume |
| FCM_SENDS_TOTAL | BIGINT | fcm_sends_total - FCM sends total |
| FCM_RECV_WAIT_TIME | BIGINT | fcm_recv_wait_time - FCM recv wait time |
| FCM_SEND_WAIT_TIME | BIGINT | fcm_send_wait_time - FCM send wait time |
| IPC_RECV_VOLUME | BIGINT | ipc_recv_volume - Interprocess communication
recv volume |
| IPC_RECV_WAIT_TIME | BIGINT | ipc_recv_wait_time - Interprocess communication
recv wait time |
| IPC_RECVS_TOTAL | BIGINT | ipc_recv_total - Interprocess communication recvs
total |
| IPC_SEND_VOLUME | BIGINT | ipc_send_volume - Interprocess communication
send volume |
| IPC_SEND_WAIT_TIME | BIGINT | ipc_send_wait_time - Interprocess communication
send wait time |
| IPC_SENDS_TOTAL | BIGINT | ipc_sends_total - Interprocess communication send
total |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOG_BUFFER_WAIT_TIME | BIGINT | log_buffer_wait_time - Log buffer wait time |
| NUM_LOG_BUFFER_FULL | BIGINT | num_log_buffer_full - Number of full log buffers |
| LOG_DISK_WAIT_TIME | BIGINT | log_disk_wait_time - Log disk wait time |
| LOG_DISK_WAITS_TOTAL | BIGINT | log_disk_waits_total - Log disk waits total |
| NUM_LOCKS_HELD | BIGINT | locks_held - Locks held |
| RQSTS_COMPLETED_TOTAL | BIGINT | rqsts_completed_total - Total requests completed |
| ROWS_MODIFIED | BIGINT | rows_modified - Rows modified |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_RETURNED | BIGINT | rows_returned - Rows returned |
| TCPIP_RECV_VOLUME | BIGINT | tcpip_recv_volume - TCP/IP received volume |
| TCPIP_SEND_VOLUME | BIGINT | tcpip_send_volume - TCP/IP send volume |
| TCPIP_RECV_WAIT_TIME | BIGINT | tcpip_recv_wait_time - TCP/IP recv wait time |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|----------------------------------|-----------|---|
| TCPIP_RECVS_TOTAL | BIGINT | tcpip_recv_total - TCP/IP recvs total |
| TCPIP_SEND_WAIT_TIME | BIGINT | tcpip_send_wait_time - TCP/IP send wait time |
| TCPIP_SENDS_TOTAL | BIGINT | tcpip_sends_total - TCP/IP sends total |
| TOTAL_APP_RQST_TIME | BIGINT | total_app_rqst_time - Total application request time |
| TOTAL_RQST_TIME | BIGINT | total_rqst_time - Total request time |
| WLM_QUEUE_TIME_TOTAL | BIGINT | wlm_queue_time_total - Workload manager total queue time |
| WLM_QUEUE_ASSIGNMENTS_TOTAL | BIGINT | wlm_queue_assignments_total - Workload manager total queue assignments |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time |
| TOTAL_WAIT_TIME | BIGINT | total_wait_time - Total wait time |
| APP_RQSTS_COMPLETED_TOTAL | BIGINT | app_rqsts_completed_total - Total application requests completed |
| TOTAL_SECTION_SORT_TIME | BIGINT | total_section_sort_time - Total section sort time |
| TOTAL_SECTION_SORT_PROC_TIME | BIGINT | total_section_sort_proc_time - Total section sort processing time |
| TOTAL_SECTION_SORTS | BIGINT | total_section_sorts - Total section sorts |
| TOTAL_SORTS | BIGINT | total_sorts - Total Sorts |
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post threshold sorts |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post shared threshold sorts |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| TOTAL_COMPILE_TIME | BIGINT | total_compile_time - Total compile time |
| TOTAL_COMPILE_PROC_TIME | BIGINT | total_compile_proc_time - Total compile processing time |
| TOTAL_COMPILATIONS | BIGINT | total_compilations - Total compilations |
| TOTAL_IMPLICIT_COMPILE_TIME | BIGINT | total_implicit_compile_time - Total implicit compile time |
| TOTAL_IMPLICIT_COMPILE_PROC_TIME | BIGINT | total_implicit_compile_proc_time - Total implicit compile processing time |
| TOTAL_IMPLICIT_COMPILATIONS | BIGINT | total_implicit_compilations - Total implicit complications |
| TOTAL_SECTION_TIME | BIGINT | total_section_time - Total section time |
| TOTAL_SECTION_PROC_TIME | BIGINT | total_section_proc_time - Total section processing time |
| TOTAL_APP_SECTION_EXECUTIONS | BIGINT | total_app_section_executions - Total section executions |
| TOTAL_ACT_TIME | BIGINT | total_act_time - Total activity time |
| TOTAL_ACT_WAIT_TIME | BIGINT | total_act_wait_time - Total activity wait time |
| ACT_RQSTS_TOTAL | BIGINT | act_rqsts_total - Total activity requests |
| TOTAL_ROUTINE_TIME | BIGINT | total_routine_time - Total routine time |
| TOTAL_ROUTINE_INVOCATIONS | BIGINT | total_routine_invocations - Total routine invocations |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|--------------------------------|-----------|---|
| TOTAL_COMMIT_TIME | BIGINT | total_commit_time - Total commit time |
| TOTAL_COMMIT_PROC_TIME | BIGINT | total_commit_proc_time - Total commits processing time |
| TOTAL_APP_COMMITS | BIGINT | total_app_commits - Total application commits |
| INT_COMMITS | BIGINT | int_commits - Internal commits |
| TOTAL_ROLLBACK_TIME | BIGINT | total_rollback_time - Total rollback time |
| TOTAL_ROLLBACK_PROC_TIME | BIGINT | total_rollback_proc_time - Total rollback processing time |
| TOTAL_APP_ROLLBACKS | BIGINT | total_app_rollback - Total application rollbacks |
| INT_ROLLBACKS | BIGINT | int_rollback - Internal rollbacks |
| TOTAL_RUNSTATS_TIME | BIGINT | total_runstats_time - Total runtime statistics |
| TOTAL_RUNSTATS_PROC_TIME | BIGINT | total_runstats_proc_time - Total runtime statistics processing time |
| TOTAL_RUNSTATS | BIGINT | total_runstats - Total runtime statistics |
| TOTAL_REORG_TIME | BIGINT | total_reorg_time - Total reorganization time |
| TOTAL_REORG_PROC_TIME | BIGINT | total_reorg_proc_time - Total reorganization processing time |
| TOTAL_REORGS | BIGINT | total_reorgs - Total reorganizations |
| TOTAL_LOAD_TIME | BIGINT | total_load_time - Total load time |
| TOTAL_LOAD_PROC_TIME | BIGINT | total_load_proc_time - Total load processing time |
| TOTAL_LOADS | BIGINT | total_loads - Total loads |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups |
| THRESH_VIOLATIONS | BIGINT | thresh_violations - Number of threshold violations |
| NUM_LW_THRESH_EXCEEDED | BIGINT | num_lw_thresh_exceeded - Number of thresholds exceeded |
| UOW_LOG_SPACE_USED | BIGINT | uow_log_space_used - Unit of Work Log Space Used |
| LOCK_WAITS_GLOBAL | BIGINT | lock_waits_global - Lock waits global |
| LOCK_WAIT_TIME_GLOBAL | BIGINT | lock_wait_time_global - Lock wait time global |
| LOCK_TIMEOUTS_GLOBAL | BIGINT | lock_timeouts_global - Lock timeouts global |
| LOCK_ESCALS_MAXLOCKS | BIGINT | lock_escals_maxlocks - Number of maxlocks lock escalations |
| LOCK_ESCALS_LOCKLIST | BIGINT | lock_escals_locklist - Number of locklist lock escalations |
| LOCK_ESCALS_GLOBAL | BIGINT | lock_escals_global - Number of global lock escalations |
| RECLAIM_WAIT_TIME | BIGINT | reclaim_wait_time - Reclaim wait time |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME | BIGINT | spacemappage_reclaim_wait_time - Space map page reclaim wait time |
| CF_WAITS | BIGINT | cf_waits - Number of cluster caching facility waits |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|------------------------------|--------------|--|
| CF_WAIT_TIME | BIGINT | cf_wait_time - cluster caching facility wait time |
| POOL_DATA_GBP_L_READS | BIGINT | pool_data_gbp_l_reads - Group buffer pool data logical reads |
| POOL_DATA_GBP_P_READS | BIGINT | pool_data_gbp_p_reads - Group buffer pool data physical reads |
| POOL_DATA_LBP_PAGES_FOUND | BIGINT | pool_data_lbp_pages_found - Local buffer pool found data pages |
| POOL_DATA_GBP_INVALID_PAGES | BIGINT | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages |
| POOL_INDEX_GBP_L_READS | BIGINT | pool_index_gbp_l_reads - Group buffer pool index logical reads |
| POOL_INDEX_GBP_P_READS | BIGINT | pool_index_gbp_p_reads - Group buffer pool index physical reads |
| POOL_INDEX_LBP_PAGES_FOUND | BIGINT | pool_index_lbp_pages_found - Local buffer pool index pages found |
| POOL_INDEX_GBP_INVALID_PAGES | BIGINT | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages |
| POOL_XDA_GBP_L_READS | BIGINT | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |
| POOL_XDA_GBP_P_READS | BIGINT | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |
| POOL_XDA_LBP_PAGES_FOUND | BIGINT | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |
| POOL_XDA_GBP_INVALID_PAGES | BIGINT | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| AUDIT_EVENTS_TOTAL | BIGINT | audit_events_total - Total audit events |
| AUDIT_FILE_WRITES_TOTAL | BIGINT | audit_file_writes_total - Total Audit files written |
| AUDIT_FILE_WRITE_WAIT_TIME | BIGINT | audit_file_write_wait_time - Audit file write wait time |
| AUDIT_SUBSYSTEM_WAITS_TOTAL | BIGINT | audit_subsystem_waits_total - Total audit subsystem waits |
| AUDIT_SUBSYSTEM_WAIT_TIME | BIGINT | audit_subsystem_wait_time - Audit subsystem wait time |
| CLIENT_HOSTNAME | VARCHAR(255) | client_hostname - Client hostname monitor element |
| CLIENT_PORT_NUMBER | INTEGER | client_port_number - Client port number monitor element |
| DIAGLOG_WRITES_TOTAL | BIGINT | diaglog_writes_total - Diag log total writes |
| DIAGLOG_WRITE_WAIT_TIME | BIGINT | diaglog_write_wait_time - Diag log write time |
| FCM_MESSAGE_RECVS_TOTAL | BIGINT | fcm_message_recvs_total - FCM message recvs total |
| FCM_MESSAGE_RECV_VOLUME | BIGINT | fcm_message_recv_volume - FCM message recv volume |
| FCM_MESSAGE_RECV_WAIT_TIME | BIGINT | fcm_message_recv_wait_time - FCM message recv wait time |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|-----------------------------------|--------------------------|--|
| FCM_MESSAGE_SENDS_TOTAL | BIGINT | fcm_message_sends_total - FCM message sends total |
| FCM_MESSAGE_SEND_VOLUME | BIGINT | fcm_message_send_volume - FCM message send volume |
| FCM_MESSAGE_SEND_WAIT_TIME | BIGINT | fcm_message_send_wait_time - FCM message send wait time |
| FCM_TQ_RECVS_TOTAL | BIGINT | fcm_tq_recvs_total - FCM tablequeue recvs total |
| FCM_TQ_RECV_VOLUME | BIGINT | fcm_tq_recv_volume - FCM tablequeue recv volume |
| FCM_TQ_RECV_WAIT_TIME | BIGINT | fcm_tq_recv_wait_time - FCM tablequeue recv wait time |
| FCM_TQ_SENDS_TOTAL | BIGINT | fcm_tq_sends_total - FCM tablequeue send total |
| FCM_TQ_SEND_VOLUME | BIGINT | fcm_tq_send_volume - FCM tablequeue send volume |
| FCM_TQ_SEND_WAIT_TIME | BIGINT | fcm_tq_send_wait_time - FCM tablequeue send wait time |
| LAST_EXECUTABLE_ID | VARCHAR(32) FOR BIT DATA | last_executable_id - Last executable identifier |
| LAST_REQUEST_TYPE | VARCHAR(32) | last_request_type - Last request type |
| TOTAL_ROUTINE_USER_CODE_PROC_TIME | BIGINT | total_routine_user_code_proc_time - Total routine user code processing time |
| TOTAL_ROUTINE_USER_CODE_TIME | BIGINT | total_routine_user_code_time - Total routine user code time |
| TQ_TOT_SEND_SPILLS | BIGINT | tq_tot_send_spills - Total number of table queue buffers overflowed |
| EVMON_WAIT_TIME | BIGINT | evmon_wait_time - Event monitor wait time |
| EVMON_WAITS_TOTAL | BIGINT | evmon_waits_total - Event monitor total waits |
| TOTAL_EXTENDED_LATCH_WAIT_TIME | BIGINT | total_extended_latch_wait_time - Total extended latch wait time |
| TOTAL_EXTENDED_LATCH_WAITS | BIGINT | total_extended_latch_waits - Total extended latch waits |
| INTRA_PARALLEL_STATE | VARCHAR(3) | intra_parallel_state - Currentstate of intrapartition parallelism |
| TOTAL_STATS_FABRICATION_TIME | BIGINT | total_stats_fabrication_time - Total statistics fabrication time |
| TOTAL_STATS_FABRICATION_PROC_TIME | BIGINT | total_stats_fabrication_proc_time - Total statistics fabrication processing time |
| TOTAL_STATS_FABRICATIONS | BIGINT | total_stats_fabrications - Total statistics fabrications |
| TOTAL_SYNC_RUNSTATS_TIME | BIGINT | total_sync_runstats_time - Total synchronous RUNSTATS time |
| TOTAL_SYNC_RUNSTATS_PROC_TIME | BIGINT | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time |
| TOTAL_SYNC_RUNSTATS | BIGINT | total_sync_runstats - Total synchronous RUNSTATS activities |
| TOTAL_DISP_RUN_QUEUE_TIME | BIGINT | total_disp_run_queue_time - Total dispatcher run queue time |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|------------------------------------|-----------|--|
| TOTAL_PEDS | BIGINT | total_peds - Total partial early distincts |
| DISABLED_PEDS | BIGINT | disabled_peds - Disabled partial early distincts |
| POST_THRESHOLD_PEDS | BIGINT | post_threshold_peds - Partial early distincts threshold |
| TOTAL_PEAS | BIGINT | total_peas - Total partial early aggregations |
| POST_THRESHOLD_PEAS | BIGINT | post_threshold_peas - Partial early aggregation threshold |
| TQ_SORT_HEAP_REQUESTS | BIGINT | tq_sort_heap_requests - Table queue sort heap requests |
| TQ_SORT_HEAP_REJECTIONS | BIGINT | tq_sort_heap_rejections - Table queue sort heap rejections |
| POOL_QUEUED_ASYNC_DATA_REQS | BIGINT | pool_queued_async_data_reqs - Data prefetch requests monitor element |
| POOL_QUEUED_ASYNC_INDEX_REQS | BIGINT | pool_queued_async_index_reqs - Index prefetch requests monitor element |
| POOL_QUEUED_ASYNC_XDA_REQS | BIGINT | pool_queued_async_xda_reqs - XDA prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_OTHER_REQS | BIGINT | pool_queued_async_other_reqs - Non-prefetch requests monitor element |
| POOL_QUEUED_ASYNC_DATA_PAGES | BIGINT | pool_queued_async_data_pages - Data pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_INDEX_PAGES | BIGINT | pool_queued_async_index_pages - Index pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_XDA_PAGES | BIGINT | pool_queued_async_xda_pages - XDA pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES | BIGINT | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES | BIGINT | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_DATA_REQS | BIGINT | pool_failed_async_data_reqs - Failed data prefetch requests monitor element |
| POOL_FAILED_ASYNC_INDEX_REQS | BIGINT | pool_failed_async_index_reqs - Failed index prefetch requests monitor element |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|--|
| POOL_FAILED_ASYNC_XDA_REQS | BIGINT | pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element |
| POOL_FAILED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_OTHER_REQS | BIGINT | pool_failed_async_other_reqs - Failed non-prefetch requests monitor element |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| PREFETCH_WAITS | BIGINT | prefetch_waits - Prefetcher wait count monitor element |
| APP_ACT_COMPLETED_TOTAL | BIGINT | app_act_completed_total - Total successful external coordinator activities monitor element |
| APP_ACT_ABORTED_TOTAL | BIGINT | app_act_aborted_total - Total failed external coordinator activities monitor element |
| APP_ACT_REJECTED_TOTAL | BIGINT | app_act_rejected_total - Total rejected external coordinator activities monitor element |
| TOTAL_CONNECT_REQUEST_TIME | BIGINT | total_connect_request_time - Total connection or switch user request time monitor element |
| TOTAL_CONNECT_REQUEST_PROC_TIME | BIGINT | total_connect_request_proc_time - Total connection or switch user request processing time monitor element |
| TOTAL_CONNECT_REQUESTS | BIGINT | total_connect_requests - Connection or switch user requests monitor element |
| TOTAL_CONNECT_AUTHENTICATION_TIME | BIGINT | total_connect_authentication_time - Total connection or switch user authentication request time monitor element |
| TOTAL_CONNECT_AUTHENTICATION_PROC_TIME | BIGINT | total_connect_authentication_proc_time - Total connection authentication processing time monitor element |
| TOTAL_CONNECT_AUTHENTIFICATIONS | BIGINT | total_connect_authentications - Connections or switch user authentications performed monitor element |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |
| COMM_EXIT_WAIT_TIME | BIGINT | comm_exit_wait_time - Communication buffer exit wait time monitor element |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|------------------------------|-----------|--|
| COMM_EXIT_WAITS | BIGINT | comm_exit_waits - Communication buffer exit number of waits monitor element |
| IDA_SEND_WAIT_TIME | BIGINT | The total amount of time spent waiting to send data to an in-database analytics process. |
| IDA_SENDS_TOTAL | BIGINT | The total number of times data was sent to an in-database analytics process. |
| IDA_SEND_VOLUME | BIGINT | The total volume of data sent from the database server to an in-database analytics process. The value is reported in bytes. |
| IDA_RECV_WAIT_TIME | BIGINT | The total amount of time spent waiting to receive data from an in-database analytics process. |
| IDA_RECVS_TOTAL | BIGINT | The total number of times data was received from an in-database analytics process. |
| IDA_RECV_VOLUME | BIGINT | The total volume of data the database server received from an in-database analytics process. The value is reported in bytes. |
| MEMBER_SUBSET_ID | INTEGER | member_subset_id - Member subset ID |
| IS_SYSTEM_APPL | SMALLINT | is_system_appl - Is System Application monitor element |
| LOCK_TIMEOUT_VAL | BIGINT | lock_timeout_val - Lock timeout value monitor element |
| CURRENT_ISOLATION | CHAR(2) | current_isolation - Current isolation level |
| NUM_LOCKS_WAITING | BIGINT | locks_waiting - Current agents waiting on locks monitor element |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted monitor element |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted monitor element |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated monitor element |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total Hash Joins monitor element |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total Hash Loops monitor element |
| HASH_JOIN_OVERFLOWS | BIGINT | hash_join_overflows - Hash Join Overflows monitor element |
| HASH_JOIN_SMALL_OVERFLOWS | BIGINT | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| POST_SHRTHRESHOLD_HASH_JOINS | BIGINT | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP Functions monitor element |
| OLAP_FUNC_OVERFLOWS | BIGINT | olap_func_overflows - OLAP Function Overflows monitor element |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element |
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL Statements Attempted monitor element |
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed Statement Operations monitor element |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---------------------------------------|-----------|---|
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL Statements Executed monitor element |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| MERGE_SQL_STMTS | BIGINT | merge_sql_stmts - Merge SQL statements executed |
| XQUERY_STMTS | BIGINT | xquery_stmts - XQuery Statements Attempted monitor element |
| IMPLICIT_REBINDS | BIGINT | implicit_rebinds - number of implicit rebinds |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/Precompiles Attempted monitor element |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal Rows Inserted monitor element |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal Rows Updated monitor element |
| CALL_SQL_STMTS | BIGINT | call_sql_stmts - CALL SQL statements executed |
| POOL_COL_L_READS | BIGINT | pool_col_l_reads - Buffer pool column-organized logical reads |
| POOL_TEMP_COL_L_READS | BIGINT | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| POOL_COL_P_READS | BIGINT | pool_col_p_reads - Buffer pool column-organized physical reads |
| POOL_TEMP_COL_P_READS | BIGINT | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| POOL_COL_LBP_PAGES_FOUND | BIGINT | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found |
| POOL_COL_WRITES | BIGINT | pool_col_writes - Buffer pool column-organized writes |
| POOL_COL_GBP_L_READS | BIGINT | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| POOL_COL_GBP_P_READS | BIGINT | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| POOL_COL_GBP_INVALID_PAGES | BIGINT | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |
| POOL_QUEUED_ASYNC_COL_REQS | BIGINT | pool_queued_async_col_reqs - Column-organized prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_REQS | BIGINT | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| POOL_QUEUED_ASYNC_COL_PAGES | BIGINT | pool_queued_async_col_pages - Column-organized page prefetch requests |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|--------------------------------------|--------------|---|
| POOL_QUEUED_ASYNC_TEMP_COL_PAGES | BIGINT | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| POOL_FAILED_ASYNC_COL_REQS | BIGINT | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| POOL_FAILED_ASYNC_TEMP_COL_REQS | BIGINT | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |
| TOTAL_COL_TIME | BIGINT | total_col_time - Total column-organized time |
| TOTAL_COL_PROC_TIME | BIGINT | total_col_proc_time - Total column-organized processing time |
| TOTAL_COL_EXECUTIONS | BIGINT | total_col_executions - Total column-organized executions |
| CLIENT_IPADDR | VARCHAR(128) | client_ipaddr - Client IP address monitor element |
| SQL_REQS_SINCE_COMMIT | BIGINT | sql_reqs_since_commit - SQL Requests Since Last Commit monitor element |
| UOW_STOP_TIME | TIMESTAMP | uow_stop_time - Unit of work stop timestamp monitor element |
| PREV_UOW_STOP_TIME | TIMESTAMP | prev_uow_stop_time - Previous Unit of Work Completion Timestamp monitor element |
| UOW_COMP_STATUS | VARCHAR(14) | uow_comp_status - Unit of Work Completion Status monitor element |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of Associated Agents monitor element |
| POST_THRESHOLD_HASH_JOINS | BIGINT | post_threshold_hash_joins - Hash Join Threshold monitor element |
| POOL_DATA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|--------------|--|
| POOL_COL_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_READ_TIME | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_WRITE_TIME | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| TOTAL_HASH_GRPBYS | BIGINT | total_hash_grpbys - Total hash group by operations |
| HASH_GRPBY_OVERFLOWES | BIGINT | hash_grpby_overflows - Hash group by overflows |
| POST_THRESHOLD_HASH_GRPBYS | BIGINT | post_threshold_hash_grpbys - Hash group by threshold |
| EXECUTION_ID | VARCHAR(128) | execution_id - User Login ID monitor element |
| POST_THRESHOLD_OLAP_FUNCS | BIGINT | post_threshold_olap_funcs - OLAP function threshold monitor element |
| POST_THRESHOLD_COL_VECTOR_CONSUMERS | BIGINT | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| TOTAL_COL_VECTOR_CONSUMERS | BIGINT | total_col_vector_consumers - Total columnar_vector consumers |
| ACTIVE_HASH_GRPBYS | BIGINT | active_hash_grpbys - Active hash GROUP BY operations |
| ACTIVE_HASH_JOINS | BIGINT | active_hash_joins - Active hash joins |
| ACTIVE_OLAP_FUNCS | BIGINT | active_olap_funcs - Active OLAP Functions |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|------------------------------|-----------|--|
| ACTIVE_PEAS | BIGINT | active_peas - Active partial early aggregations operations |
| ACTIVE_PEDS | BIGINT | active_peds - Active partial early distinct operations |
| ACTIVE_SORT_CONSUMERS | BIGINT | active_sort_consumers - Active sort consumers |
| ACTIVE_SORTS | BIGINT | active_sorts - Active Sorts |
| ACTIVE_COL_VECTOR_CONSUMERS | BIGINT | active_col_vector_consumers - Active columnar_vector consumers |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total Sort Heap Allocated |
| SORT_SHRHEAP_ALLOCATED | BIGINT | sort_shrheap_allocated - Sort Share Heap Currently Allocated |
| TOTAL_BACKUP_TIME | BIGINT | total_backup_time - Total elapsed time for online backups |
| TOTAL_BACKUP_PROC_TIME | BIGINT | total_backup_proc_time - Total non-wait time for online backups |
| TOTAL_BACKUPS | BIGINT | total_backups - Total backups |
| TOTAL_INDEX_BUILD_TIME | BIGINT | total_index_build_time - Total time spent building indexes due to index creation, recreation or rebuilding |
| TOTAL_INDEX_BUILD_PROC_TIME | BIGINT | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation, recreation or rebuilding |
| TOTAL_INDEXES_BUILT | BIGINT | total_indexes_built - Total number of indexes built |
| FCM_TQ_RECV_WAITS_TOTAL | BIGINT | fcm_tq_recv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| FCM_MESSAGE_RECV_WAITS_TOTAL | BIGINT | fcm_message_recv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| FCM_TQ_SEND_WAITS_TOTAL | BIGINT | fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |
| FCM_MESSAGE_SEND_WAITS_TOTAL | BIGINT | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| FCM_SEND_WAITS_TOTAL | BIGINT | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| FCM_RECV_WAITS_TOTAL | BIGINT | fcm_recv_waits_total - Number of times spent waiting to receive data through FCM monitor element |
| EXT_TABLE_RECV_WAIT_TIME | BIGINT | ext_table_recv_wait_time - Total agent wait time for external table readers monitor element |
| EXT_TABLE_RECVS_TOTAL | BIGINT | ext_table_recvs_total - Total row batches received from external table readers monitor element |
| EXT_TABLE_RECV_VOLUME | BIGINT | ext_table_recv_volume - Total data received from external table readers monitor element |
| EXT_TABLE_READ_VOLUME | BIGINT | ext_table_read_volume - Total data read by external table readers monitor element |

Table 168. Information returned for MON_GET_UNIT_OF_WORK (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|-------------------------------|-----------|--|
| EXT_TABLE_SEND_WAIT_TIME | BIGINT | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| EXT_TABLE_SENDS_TOTAL | BIGINT | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| EXT_TABLE_SEND_VOLUME | BIGINT | ext_table_send_volume - Total data sent to external table writers monitor element |
| EXT_TABLE_WRITE_VOLUME | BIGINT | ext_table_write_volume - Total data written by external table writers monitor element |
| COL_VECTOR_CONSUMER_OVERFLOW | BIGINT | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |
| TOTAL_COL_SYNOPSIS_TIME | BIGINT | total_col_synopsis_time - Total column-organized synopsis table time monitor element |
| TOTAL_COL_SYNOPSIS_PROC_TIME | BIGINT | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| TOTAL_COL_SYNOPSIS_EXECUTIONS | BIGINT | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |
| COL_SYNOPSIS_ROWS_INSERTED | BIGINT | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| LOB_PREFETCH_WAIT_TIME | BIGINT | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| LOB_PREFETCH_REQS | BIGINT | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| FED_ROWS_DELETED | BIGINT | fed_rows_deleted - Rows deleted by a federation system monitor element |
| FED_ROWS_INSERTED | BIGINT | fed_rows_inserted - Rows inserted by a federation system monitor element |
| FED_ROWS_UPDATED | BIGINT | fed_rows_updated - Rows updated by a federation system monitor element |
| FED_ROWS_READ | BIGINT | fed_rows_read - Rows read by a federation system monitor element |
| FED_WAIT_TIME | BIGINT | fed_wait_time - Time spent by a federation server monitor element |
| FED_WAITS_TOTAL | BIGINT | fed_waits_total - Total number of execution times for a federation server monitor element |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Insert count for SQL workspace monitor element |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section lookups - SQL workspace access count monitor element |
| ADM_OVERFLOW | BIGINT | Reserved for future use. |
| ADM_BYPASS_ACT_TOTAL | BIGINT | Reserved for future use. |

Usage notes

The metrics returned by the MON_GET_UNIT_OF_WORK table function represent the accumulation of all metrics for requests that were submitted during a unit of

work. Metrics are rolled up periodically during the unit of work. Therefore, the values reported by this table function reflect the current state of the system at the time of the most recent rollup. Metrics are strictly increasing in value. To determine the value of a given metric for an interval of time, use the function to query the metric at the start and end of the interval, and compute the difference.

Request metrics are controlled through the COLLECT REQUEST METRICS clause on service superclasses and the *mon_req_metrics* database configuration parameter at the database level. Metrics are only collected for a request if the request is processed by an agent in a service subclass whose parent service superclass has request metrics enabled, or if request metrics collection is enabled for the entire database. By default, request metrics are enabled at the database level. If request metrics have been disabled at the database level, and for a service superclass, the metrics reported for each unit of work mapped to that service superclass stop increasing (or remain at 0 if request metrics were disabled at database activation time).

The MON_GET_UNIT_OF_WORK table function returns one row of data per unit of work and per member. No aggregation across units of work (on a member), or across members (for a service class or more), is performed. However, aggregation can be achieved through SQL queries. The input parameters have the effect of being ANDed together.

Example

Identify the units of work that are consuming the highest amount of CPU time on the system.

```
SELECT application_handle,
       uow_id,
       total_cpu_time,
       app_rqsts_completed_total,
       rqsts_completed_total
FROM TABLE(MON_GET_UNIT_OF_WORK(NULL,-1)) AS t
ORDER BY total_cpu_time DESC
```

The following is an example of output from this query.

| APPLICATION_HANDLE | UOW_ID | TOTAL_CPU_TIME | ... |
|--------------------|--------|----------------|-----------|
| ----- | ----- | ----- | ... |
| | 46 | 5 | 27959 ... |

1 record(s) selected.

Output for query (continued).

| ... | APP_RQSTS_COMPLETED_TOTAL | RQSTS_COMPLETED_TOTAL |
|-----|---------------------------|-----------------------|
| ... | ----- | ----- |
| ... | | 72 48 |

MON_GET_UNIT_OF_WORK_DETAILS table function - Get unit of work metrics as an XML document

The MON_GET_UNIT_OF_WORK_DETAILS table function returns metrics for one or more units of work.

The metrics are returned in an XML document.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

```
►► MON_GET_UNIT_OF_WORK_DETAILS—(—————►  
► application_handle—, —member— [—————, —system_appls—] )—————►
```

The schema is SYSPROC.

Routine parameters

application_handle

An input argument of type BIGINT that specifies a valid application handle in the same database as the one currently connected to when calling this function. If the argument is null, metrics are retrieved for units of work running in all superclasses in the database.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is set implicitly.

system_appls

An input parameter of data type SMALLINT that specifies whether information for system applications is returned. The following values are valid:

- 0 or NULL: System application information is not returned
- 1: User and system application information is returned

Information returned

Table 169. Information returned for MON_GET_UNIT_OF_WORK_DETAILS

| Column Name | Data Type | Description or corresponding monitor element |
|-------------------------|--------------|--|
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name |
| SERVICE_SUBCLASS_NAME | VARCHAR(128) | service_subclass_name - Service subclass name |
| SERVICE_CLASS_ID | INTEGER | service_class_id - Service class ID |
| MEMBER | SMALLINT | member - Database member |
| COORD_MEMBER | SMALLINT | coord_member - Coordinator member Database member for the coordinator partition of the given unit of work. |

Table 169. Information returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|------------------------|--------------|--|
| APPLICATION_HANDLE | BIGINT | application_handle - Application handle |
| WORKLOAD_NAME | VARCHAR(128) | workload_name - Workload name |
| WORKLOAD_OCCURRENCE_ID | INTEGER | workload_occurrence_id - Workload occurrence identifier. This ID does not uniquely identify the workload occurrence unless it is coupled with the coordinator database partition number and the workload name. |
| UOW_ID | INTEGER | uow_id - Unit of work ID |
| DETAILS | BLOB(1M) | XML document that contains detailed metrics for the unit of work. See Table 170 for a description of the elements in this document. |

The following example shows the structure of the XML document that is returned in the DETAILS column.

```
<db2_unit_of_work xmlns="http://www.ibm.com/xmlns/prod/db2/mon" release="90700000">
  <service_superclass_name>SYSDEFAULTUSERCLASS</service_superclass_name>
  <service_subclass_name>SYSDEFAULTSUBCLASS</service_subclass_name>
  <service_class_id>13</service_class_id>
  <workload_name>SYSDEFAULTUSERWORKLOAD</workload_name>
  <member>0</member>
  <coord_member>0</coord_member>
  <application_handle>21</application_handle>
  <workload_occurrence_id>1</workload_occurrence_id>
  <uow_id>2</uow_id>
  <workload_occurrence_state>UOWEXEC</workload_occurrence_state>
  <system_metrics>
    <act_aborted_total>5</act_aborted_total>
    ...
    <wlm_queue_assignments_total>3</wlm_queue_assignments_total>
  </system_metrics>
</db2_unit_of_work_metrics>
```

For the full schema, see sqllib/misc/DB2MonRoutines.xsd.

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS

| Element Name | Data Type | Description or corresponding monitor element |
|-----------------------------|-----------------------|--|
| act_aborted_total | xs:nonNegativeInteger | act_aborted_total - Total aborted activities |
| act_completed_total | xs:nonNegativeInteger | act_completed_total - Total completed activities |
| act_rejected_total | xs:nonNegativeInteger | act_rejected_total - Total rejected activities |
| act_rqsts_total | xs:nonNegativeInteger | act_rqsts_total - Total activity requests |
| active_col_vector_consumers | xs:long | active_col_vector_consumers - Active columnar_vector consumers |
| active_hash_grpbys | xs:long | active_hash_grpbys - Active hash GROUP BY operations |
| active_hash_joins | xs:long | active_hash_joins - Active hash joins |
| active_olap_funcs | xs:long | active_olap_funcs - Active OLAP Functions |
| active_peas | xs:long | active_peas - Active partial early aggregations operations |
| active_peds | xs:long | active_peds - Active partial early distinct operations |
| active_sort_consumers | xs:long | active_sort_consumers - Active sort consumers |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|-----------------------------|-----------------------|--|
| active_sorts | xs:long | active_sorts - Active Sorts |
| adm_bypass_act_total | xs:long | Reserved for future use. |
| adm_overflows | xs:long | Reserved for future use. |
| agent_wait_time | xs:nonNegativeInteger | agent_wait_time - Agent wait time |
| agent_waits_total | xs:nonNegativeInteger | agent_waits_total - Total agent waits |
| app_act_aborted_total | xs:nonNegativeInteger | app_act_aborted_total - Total failed external coordinator activities monitor element |
| app_act_completed_total | xs:nonNegativeInteger | app_act_completed_total - Total successful external coordinator activities monitor element |
| app_act_rejected_total | xs:nonNegativeInteger | app_act_rejected_total - Total rejected external coordinator activities monitor element |
| app_rqsts_completed_total | xs:nonNegativeInteger | app_rqsts_completed_total - Total application requests completed |
| appl_section_inserts | xs:nonNegativeInteger | appl_section_inserts - Insert count for SQL workspace monitor element |
| appl_section_lookups | xs:nonNegativeInteger | appl_section_lookups - SQL workspace access count monitor element |
| application_handle | xs:nonNegativeInteger | application_handle - Application handle |
| application_id | xs:string | appl_id - Application ID |
| audit_events_total | xs:nonNegativeInteger | audit_events_total - Total audit events |
| audit_file_write_wait_time | xs:nonNegativeInteger | audit_file_write_wait_time - Audit file write wait time |
| audit_file_writes_total | xs:nonNegativeInteger | audit_file_writes_total - Total Audit files written |
| audit_subsystem_wait_time | xs:nonNegativeInteger | audit_subsystem_wait_time - Audit subsystem wait time |
| audit_subsystem_waits_total | xs:nonNegativeInteger | audit_subsystem_waits_total - Total audit subsystem waits |
| binds_precompiles | xs:long | binds_precompiles - Binds/Precompiles Attempted monitor element |
| call_sql_stmts | xs:long | call_sql_stmts - CALL SQL statements executed |
| cat_cache_inserts | xs:nonNegativeInteger | cat_cache_inserts - Catalog cache inserts |
| cat_cache_lookups | xs:nonNegativeInteger | cat_cache_lookups - Catalog cache lookups |
| cf_wait_time | xs:long | cf_wait_time - cluster caching facility wait time |
| cf_waits | xs:long | cf_waits - Number of cluster caching facility waits |
| client_acctng | xs:string | client_acctng - Client accounting string monitor element |
| client_applname | xs:string | client_applname - Client application name monitor element |
| client_hostname | xs:string | client_hostname - Client hostname |
| client_idle_wait_time | xs:nonNegativeInteger | client_idle_wait_time - Client idle wait time |
| client_ipaddr | xs:string | client_ipaddr - Client IP address monitor element |
| client_port_number | xs:nonNegativeInteger | client_port_number - Client port number |
| client_userid | xs:string | client_userid - Client user ID monitor element |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|-------------------------------|-----------------------|---|
| client_wrkstnname | xs:string | client_wrkstnname - Client workstation name monitor element |
| col_synopsis_rows_inserted | xs:nonNegativeInteger | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| col_vector_consumer_overflows | xs:long | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |
| comm_exit_wait_time | xs:nonNegativeInteger | comm_exit_wait_time - Communication buffer exit wait time monitor element |
| comm_exit_waits | xs:nonNegativeInteger | comm_exit_waits - Communication buffer exit number of waits monitor element |
| coord_member | xs:nonNegativeInteger | coord_member - Coordinator member |
| current_isolation | xs:string | current_isolation - Current isolation level |
| ddl_sql_stmts | xs:long | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| deadlocks | xs:nonNegativeInteger | deadlocks - Deadlocks detected |
| diaglog_write_wait_time | xs:nonNegativeInteger | diaglog_write_wait_time - Diag log write time |
| diaglog_writes_total | xs:nonNegativeInteger | diaglog_writes_total - Diag log total writes |
| direct_read_reqs | xs:nonNegativeInteger | direct_read_reqs - Direct read requests |
| direct_read_time | xs:nonNegativeInteger | direct_read_time - Direct read time |
| direct_reads | xs:nonNegativeInteger | direct_reads - Direct reads from database |
| direct_write_reqs | xs:nonNegativeInteger | direct_write_reqs - Direct write requests |
| direct_write_time | xs:nonNegativeInteger | direct_write_time - Direct write time |
| direct_writes | xs:nonNegativeInteger | direct_writes - Direct writes to database |
| disabled_peds | xs:long | disabled_peds - Disabled partial early distincts |
| dynamic_sql_stmts | xs:long | dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element |
| evmon_wait_time | xs:nonNegativeInteger | evmon_wait_time - Event monitor wait time |
| evmon_waits_total | xs:nonNegativeInteger | evmon_waits_total - Event monitor total waits |
| execution_id | xs:string(128) | execution_id - User Login ID monitor element |
| ext_table_read_volume | xs:nonNegativeInteger | ext_table_read_volume - Total data read by external table readers monitor element |
| ext_table_rcv_volume | xs:nonNegativeInteger | ext_table_rcv_volume - Total data received from external table readers monitor element |
| ext_table_rcv_wait_time | xs:nonNegativeInteger | ext_table_rcv_wait_time - Total agent wait time for external table readers monitor element |
| ext_table_rcvs_total | xs:nonNegativeInteger | ext_table_rcvs_total - Total row batches received from external table readers monitor element |
| ext_table_send_volume | xs:nonNegativeInteger | ext_table_send_volume - Total data sent to external table writers monitor element |
| ext_table_send_wait_time | xs:nonNegativeInteger | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| ext_table_sends_total | xs:nonNegativeInteger | ext_table_sends_total - Total row batches sent to external table writers monitor element |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|------------------------------|-----------------------|--|
| ext_table_write_volume | xs:nonNegativeInteger | ext_table_write_volume - Total data written by external table writers monitor element |
| failed_sql_stmts | xs:long | failed_sql_stmts - Failed Statement Operations monitor element |
| fcm_message_rcv_volume | xs:nonNegativeInteger | fcm_message_rcv_volume - FCM message rcv volume |
| fcm_message_rcv_wait_time | xs:nonNegativeInteger | fcm_message_rcv_wait_time - FCM message rcv wait time |
| fcm_message_rcv_waits_total | xs:nonNegativeInteger | fcm_message_rcv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| fcm_message_rcvs_total | xs:nonNegativeInteger | fcm_message_rcvs_total - FCM message rcvs total |
| fcm_message_send_volume | xs:nonNegativeInteger | fcm_message_send_volume - FCM message send volume |
| fcm_message_send_wait_time | xs:nonNegativeInteger | fcm_message_send_wait_time - FCM message send wait time |
| fcm_message_send_waits_total | xs:nonNegativeInteger | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| fcm_message_sends_total | xs:nonNegativeInteger | fcm_message_sends_total - FCM message sends total |
| fcm_rcv_volume | xs:nonNegativeInteger | fcm_rcv_volume - FCM rcv volume |
| fcm_rcv_wait_time | xs:nonNegativeInteger | fcm_rcv_wait_time - FCM rcv wait time |
| fcm_rcv_waits_total | xs:nonNegativeInteger | fcm_rcv_waits_total - Number of times spent waiting to receive data through FCM monitor element |
| fcm_rcvs_total | xs:nonNegativeInteger | fcm_rcvs_total - FCM rcvs total |
| fcm_send_volume | xs:nonNegativeInteger | fcm_send_volume - FCM send volume |
| fcm_send_wait_time | xs:nonNegativeInteger | fcm_send_wait_time - FCM send wait time |
| fcm_send_waits_total | xs:nonNegativeInteger | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| fcm_sends_total | xs:nonNegativeInteger | fcm_sends_total - FCM sends total |
| fcm_tq_rcv_volume | xs:nonNegativeInteger | fcm_tq_rcv_volume - FCM tablequeue rcv volume |
| fcm_tq_rcv_wait_time | xs:nonNegativeInteger | fcm_tq_rcv_wait_time - FCM tablequeue rcv wait time |
| fcm_tq_rcv_waits_total | xs:nonNegativeInteger | fcm_tq_rcv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| fcm_tq_rcvs_total | xs:nonNegativeInteger | fcm_tq_rcvs_total - FCM tablequeue rcvs total |
| fcm_tq_send_volume | xs:nonNegativeInteger | fcm_tq_send_volume - FCM tablequeue send volume |
| fcm_tq_send_wait_time | xs:nonNegativeInteger | fcm_tq_send_wait_time - FCM tablequeue send wait time |
| fcm_tq_sends_total | xs:nonNegativeInteger | fcm_tq_sends_total - FCM tablequeue send total |
| fed_rows_deleted | xs:nonNegativeInteger | fed_rows_deleted - Rows deleted by a federation system monitor element |
| fed_rows_inserted | xs:nonNegativeInteger | fed_rows_inserted - Rows inserted by a federation system monitor element |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|---------------------------|-----------------------|--|
| fed_rows_read | xs:nonNegativeInteger | fed_rows_read - Rows read by a federation system monitor element |
| fed_rows_updated | xs:nonNegativeInteger | fed_rows_updated - Rows updated by a federation system monitor element |
| fed_wait_time | xs:nonNegativeInteger | fed_wait_time - Time spent by a federation server monitor element |
| fed_waits_total | xs:nonNegativeInteger | fed_waits_total - Total number of execution times for a federation server monitor element |
| hash_grpby_overflows | xs:long | hash_grpby_overflows - Hash group by overflows |
| hash_join_overflows | xs:long | hash_join_overflows - Hash Join Overflows monitor element |
| hash_join_small_overflows | xs:long | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| ida_recv_volume | xs:nonNegativeInteger | ida_recv_volume - Total data volume received monitor element

The total volume of data the database server received from an in-database analytics process. The value is reported in bytes. |
| ida_recv_wait_time | xs:nonNegativeInteger | ida_recv_wait_time - Time spent waiting to receive data monitor element

The total amount of time spent waiting to receive data from an in-database analytics process. |
| ida_recvs_total | xs:nonNegativeInteger | ida_recvs_total - Number of times data received monitor element

The total number of times data was received from an in-database analytics process. |
| ida_send_volume | xs:nonNegativeInteger | ida_send_volume - Total data volume sent monitor element

The total volume of data sent from the database server to an in-database analytics process. The value is reported in bytes. |
| ida_send_wait_time | xs:nonNegativeInteger | ida_send_wait_time - Time spent waiting to send data monitor element

The total amount of time spent waiting to send data to an in-database analytics process. |
| ida_sends_total | xs:nonNegativeInteger | ida_sends_total - Number of times data sent monitor element

The total number of times data was sent to an in-database analytics process. |
| implicit_rebinds | xs:long | implicit_rebinds - number of implicit rebinds |
| int_commits | xs:nonNegativeInteger | int_commits - Internal commits |
| int_rollbacks | xs:nonNegativeInteger | int_rollbacks - Internal rollbacks |
| int_rows_deleted | xs:long | int_rows_deleted - Internal Rows Deleted monitor element |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|------------------------|-----------------------|--|
| int_rows_inserted | xs:long | int_rows_inserted - Internal Rows Inserted monitor element |
| int_rows_updated | xs:long | int_rows_updated - Internal Rows Updated monitor element |
| intra_parallel_state | xs:string | intra_parallel_state - Current state of intrapartition parallelism monitor element |
| ipc_rcv_volume | xs:nonNegativeInteger | ipc_rcv_volume - Interprocess communication rcv volume |
| ipc_rcv_wait_time | xs:nonNegativeInteger | ipc_rcv_wait_time - Interprocess communication rcv wait time |
| ipc_rcvs_total | xs:nonNegativeInteger | ipc_rcvs_total - Interprocess communication rcvs total |
| ipc_send_volume | xs:nonNegativeInteger | ipc_send_volume - Interprocess communication send volume |
| ipc_send_wait_time | xs:nonNegativeInteger | ipc_send_wait_time - Interprocess communication send wait time |
| ipc_sends_total | xs:nonNegativeInteger | ipc_sends_total - Interprocess communication send total |
| is_system_appl | xs:short | is_system_appl - Is System Application monitor element |
| last_executable_id | xs:hexBinary(32) | last_executable_id - Last executable identifier |
| last_request_type | xs:string(32) | last_request_type - Last request type |
| lob_prefetch_reqs | xs:nonNegativeInteger | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| lob_prefetch_wait_time | xs:nonNegativeInteger | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| lock_escals | xs:nonNegativeInteger | lock_escals - Number of lock escalations |
| lock_escals_global | xs:long | lock_escals_global - Number of global lock escalations |
| lock_escals_locklist | xs:long | lock_escals_locklist - Number of locklist lock escalations |
| lock_escals_maxlocks | xs:long | lock_escals_maxlocks - Number of maxlocks lock escalations |
| lock_timeout_val | xs:long | lock_timeout_val - Lock timeout value monitor element |
| lock_timeouts | xs:nonNegativeInteger | lock_timeouts - Number of lock timeouts |
| lock_timeouts_global | xs:long | lock_timeouts_global - Lock timeouts global |
| lock_wait_time | xs:nonNegativeInteger | lock_wait_time - Time waited on locks |
| lock_wait_time_global | xs:long | lock_wait_time_global - Lock wait time global |
| lock_waits | xs:nonNegativeInteger | lock_waits - Lock waits |
| lock_waits_global | xs:long | lock_waits_global - Lock waits global |
| log_buffer_wait_time | xs:nonNegativeInteger | log_buffer_wait_time - Log buffer wait time |
| log_disk_wait_time | xs:nonNegativeInteger | log_disk_wait_time - Log disk wait time |
| log_disk_waits_total | xs:nonNegativeInteger | log_disk_waits_total - Log disk waits total |
| member | xs:nonNegativeInteger | member - Database member |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|--|-----------------------|--|
| member_subset_id | xs:long | member_subset_id - Member subset ID |
| merge_sql_stmts | xs:long | merge_sql_stmts - Merge SQL statements executed |
| num_assoc_agents | xs:long | num_assoc_agents - Number of Associated Agents monitor element |
| num_locks_held | xs:nonNegativeInteger | locks_held - Locks held |
| num_locks_waiting | xs:long | lock_waits - Lock waits |
| num_log_buffer_full | xs:nonNegativeInteger | num_log_buffer_full - Number of full log buffers |
| num_lw_thresh_exceeded | xs:nonNegativeInteger | num_lw_thresh_exceeded - Number of thresholds exceeded |
| olap_func_overflows | xs:long | olap_func_overflows - OLAP Function Overflows monitor element |
| pkg_cache_inserts | xs:nonNegativeInteger | pkg_cache_inserts - Package cache inserts |
| pkg_cache_lookups | xs:nonNegativeInteger | pkg_cache_lookups - Package cache lookups |
| pool_col_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |
| pool_col_gbp_invalid_pages | xs:nonNegativeInteger | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| pool_col_gbp_l_reads | xs:nonNegativeInteger | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| pool_col_gbp_p_reads | xs:nonNegativeInteger | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| pool_col_l_reads | xs:nonNegativeInteger | pool_col_l_reads - Buffer pool column-organized logical reads |
| pool_col_lbp_pages_found | xs:nonNegativeInteger | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found |
| pool_col_p_reads | xs:nonNegativeInteger | pool_col_p_reads - Buffer pool column-organized physical reads |
| pool_col_writes | xs:nonNegativeInteger | pool_col_writes - Buffer pool column-organized writes |
| pool_data_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| pool_data_gbp_invalid_pages | xs:long | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages |
| pool_data_gbp_l_reads | xs:long | pool_data_gbp_l_reads - Group buffer pool data logical reads |
| pool_data_gbp_p_reads | xs:long | pool_data_gbp_p_reads - Group buffer pool data physical reads |
| pool_data_l_reads | xs:nonNegativeInteger | pool_data_l_reads - Buffer pool data logical reads |
| pool_data_lbp_pages_found | xs:long | pool_data_lbp_pages_found - Local buffer pool found data pages |
| pool_data_p_reads | xs:nonNegativeInteger | pool_data_p_reads - Buffer pool data physical reads |
| pool_data_writes | xs:nonNegativeInteger | pool_data_writes - Buffer pool data writes |
| pool_failed_async_col_reqs | xs:nonNegativeInteger | pool_failed_async_col_reqs - Failed column-organized prefetch requests |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|---|-----------------------|--|
| pool_failed_async_data_reqs | xs:nonNegativeInteger | pool_failed_async_data_reqs - Failed data prefetch requests monitor element |
| pool_failed_async_index_reqs | xs:nonNegativeInteger | pool_failed_async_index_reqs - Failed index prefetch requests monitor element |
| pool_failed_async_temp_col_reqs | xs:nonNegativeInteger | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |
| pool_failed_async_temp_data_reqs | xs:nonNegativeInteger | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element |
| pool_failed_async_temp_index_reqs | xs:nonNegativeInteger | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element |
| pool_failed_async_temp_xda_reqs | xs:nonNegativeInteger | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element |
| pool_failed_async_other_reqs | xs:nonNegativeInteger | pool_failed_async_other_reqs - Failed non-prefetch requests monitor element |
| pool_failed_async_xda_reqs | xs:nonNegativeInteger | pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element |
| pool_index_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| pool_index_gbp_invalid_pages | xs:long | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages |
| pool_index_gbp_l_reads | xs:long | pool_index_gbp_l_reads - Group buffer pool index logical reads |
| pool_index_gbp_p_reads | xs:long | pool_index_gbp_p_reads - Group buffer pool index physical reads |
| pool_index_l_reads | xs:nonNegativeInteger | pool_index_l_reads - Buffer pool index logical reads |
| pool_index_lbp_pages_found | xs:long | pool_index_lbp_pages_found - Local buffer pool index pages found |
| pool_index_p_reads | xs:nonNegativeInteger | pool_index_p_reads - Buffer pool index physical reads |
| pool_index_writes | xs:nonNegativeInteger | pool_index_writes - Buffer pool index writes |
| pool_queued_async_col_reqs | xs:nonNegativeInteger | pool_queued_async_col_reqs - Column-organized prefetch requests |
| pool_queued_async_data_pages | xs:nonNegativeInteger | pool_queued_async_data_pages - Data pages prefetch requests monitor element |
| pool_queued_async_data_reqs | xs:nonNegativeInteger | pool_queued_async_data_reqs - Data prefetch requests monitor element |
| pool_queued_async_index_reqs | xs:nonNegativeInteger | pool_queued_async_index_reqs - Index prefetch requests monitor element |
| pool_queued_async_col_pages | xs:nonNegativeInteger | pool_queued_async_col_pages - Column-organized page prefetch requests |
| pool_queued_async_index_pages | xs:nonNegativeInteger | pool_queued_async_index_pages - Index pages prefetch requests monitor element |
| pool_queued_async_temp_col_pages | xs:nonNegativeInteger | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|---------------------------------------|-----------------------|--|
| pool_queued_async_temp_col_reqs | xs:nonNegativeInteger | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| pool_queued_async_temp_data_pages | xs:nonNegativeInteger | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_data_reqs | xs:nonNegativeInteger | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_index_pages | xs:nonNegativeInteger | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_index_reqs | xs:nonNegativeInteger | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_xda_pages | xs:nonNegativeInteger | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_xda_reqs | xs:nonNegativeInteger | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element |
| pool_queued_async_other_reqs | xs:nonNegativeInteger | pool_queued_async_other_reqs - Non-prefetch requests monitor element |
| pool_queued_async_xda_pages | xs:nonNegativeInteger | pool_queued_async_xda_pages - The number of XML storage object (XDA) data pages successfully requested for prefetching |
| pool_queued_async_xda_reqs | xs:nonNegativeInteger | pool_queued_async_xda_reqs - XDA prefetch requests monitor element |
| pool_read_time | xs:nonNegativeInteger | pool_read_time - Total buffer pool physical read time |
| pool_temp_col_l_reads | xs:nonNegativeInteger | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| pool_temp_col_p_reads | xs:nonNegativeInteger | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| pool_temp_data_l_reads | xs:nonNegativeInteger | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| pool_temp_data_p_reads | xs:nonNegativeInteger | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| pool_temp_index_l_reads | xs:nonNegativeInteger | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| pool_temp_index_p_reads | xs:nonNegativeInteger | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| pool_temp_xda_l_reads | xs:nonNegativeInteger | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |
| pool_temp_xda_p_reads | xs:nonNegativeInteger | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| pool_write_time | xs:nonNegativeInteger | pool_write_time - Total buffer pool physical write time |
| pool_xda_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|-------------------------------------|-----------------------|---|
| pool_xda_gbp_invalid_pages | xs:nonNegativeInteger | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| pool_xda_gbp_l_reads | xs:nonNegativeInteger | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |
| pool_xda_gbp_p_reads | xs:nonNegativeInteger | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |
| pool_xda_l_reads | xs:nonNegativeInteger | pool_xda_l_reads - Buffer pool XDA data logical reads |
| pool_xda_lbp_pages_found | xs:nonNegativeInteger | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |
| pool_xda_p_reads | xs:nonNegativeInteger | pool_xda_p_reads - Buffer pool XDA data physical reads |
| pool_xda_writes | xs:nonNegativeInteger | pool_xda_writes - Buffer pool XDA data writes |
| post_shrthreshold_hash_joins | xs:long | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| post_shrthreshold_sorts | xs:nonNegativeInteger | post_shrthreshold_sorts - Post shared threshold sorts |
| post_threshold_col_vector_consumers | xs:long | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| post_threshold_hash_grpbys | xs:long | post_threshold_hash_grpbys - Hash group by threshold |
| post_threshold_hash_joins | xs:long | post_threshold_hash_joins - Hash Join Threshold monitor element |
| post_threshold_olap_funcs | xs:long | post_threshold_olap_funcs - OLAP function threshold monitor element |
| post_threshold_peas | xs:long | post_threshold_peas - Partial early aggregation threshold |
| post_threshold_peds | xs:long | post_threshold_peds - Partial early distincts threshold |
| post_threshold_sorts | xs:nonNegativeInteger | post_threshold_sorts - Post threshold sorts |
| prefetch_wait_time | xs:nonNegativeInteger | prefetch_wait_time - Time waited for prefetch |
| prefetch_waits | xs:nonNegativeInteger | prefetch_waits - Prefetcher wait count monitor element |
| prev_uow_stop_time | xs:dateTime | prev_uow_stop_time - Previous Unit of Work Completion Timestamp monitor element |
| locks_waiting | xs:long | locks_waiting - Current agents waiting on locks monitor element |
| fcm_tq_send_waits_total | xs:nonNegativeInteger | fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |
| reclaim_wait_time | xs:long | reclaim_wait_time - Reclaim wait time |
| rows_deleted | xs:long | rows_deleted - Rows deleted monitor element |
| rows_inserted | xs:long | rows_inserted - Rows inserted monitor element |
| rows_modified | xs:nonNegativeInteger | rows_modified - Rows modified |
| rows_read | xs:nonNegativeInteger | rows_read - Rows read |
| rows_returned | xs:nonNegativeInteger | rows_returned - Rows returned |
| rows_updated | xs:long | rows_updated - Rows updated monitor element |
| rqsts_completed_total | xs:nonNegativeInteger | rqsts_completed_total - Total requests completed |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|--------------------------------|-----------------------|--|
| select_sql_stmts | xs:long | select_sql_stmts - Select SQL Statements Executed monitor element |
| service_class_id | xs:nonNegativeInteger | service_class_id - Service class ID |
| service_subclass_name | xs:string (128) | service_subclass_name - Service subclass name |
| service_superclass_name | xs:string (128) | service_superclass_name - Service superclass name |
| session_auth_id | xs:string | session_auth_id - Session authorization ID |
| sort_heap_allocated | xs:long | sort_heap_allocated - Total Sort Heap Allocated |
| sort_overflows | xs:nonNegativeInteger | sort_overflows - Sort overflows |
| sort_shrheap_allocated | xs:long | sort_shrheap_allocated - Sort Share Heap Currently Allocated |
| spacemappage_reclaim_wait_time | xs:long | spacemappage_reclaim_wait_time - Space map page reclaim wait time |
| sql_reqs_since_commit | xs:long | sql_reqs_since_commit - SQL Requests Since Last Commit monitor element |
| static_sql_stmts | xs:long | static_sql_stmts - Static SQL Statements Attempted monitor element |
| tcpip_rcv_volume | xs:nonNegativeInteger | tcpip_rcv_volume - TCP/IP received volume |
| tcpip_rcv_wait_time | xs:nonNegativeInteger | tcpip_rcv_wait_time - TCP/IP rcv wait time |
| tcpip_rcvs_total | xs:nonNegativeInteger | tcpip_rcvs_total - TCP/IP rcvs total |
| tcpip_send_volume | xs:nonNegativeInteger | tcpip_send_volume - TCP/IP send volume |
| tcpip_send_wait_time | xs:nonNegativeInteger | tcpip_send_wait_time - TCP/IP send wait time |
| tcpip_sends_total | xs:nonNegativeInteger | tcpip_sends_total - TCP/IP sends total |
| thresh_violations | xs:nonNegativeInteger | thresh_violations - Number of threshold violations |
| total_act_time | xs:nonNegativeInteger | total_act_time - Total activity time |
| total_act_wait_time | xs:nonNegativeInteger | total_act_wait_time - Total activity wait time |
| total_app_commits | xs:nonNegativeInteger | total_app_commits - Total application commits |
| total_app_rollbacks | xs:nonNegativeInteger | total_app_rollbacks - Total application rollbacks |
| total_app_rqst_time | xs:nonNegativeInteger | total_app_rqst_time - Total application request time |
| total_app_section_executions | xs:nonNegativeInteger | total_app_section_executions - Total section executions |
| total_backup_proc_time | xs:long | total_backup_proc_time - Total non-wait time for online backups |
| total_backup_time | xs:long | total_backup_time - Total elapsed time for online backups |
| total_backups | xs:long | total_backups - Total backups |
| total_col_executions | xs:nonNegativeInteger | total_col_executions - Total column-organized executions |
| total_col_proc_time | xs:nonNegativeInteger | total_col_proc_time - Total column-organized processing time |
| total_col_synopsis_executions | xs:nonNegativeInteger | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|--|-----------------------|--|
| total_col_synopsis_proc_time | xs:nonNegativeInteger | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| total_col_synopsis_time | xs:nonNegativeInteger | total_col_synopsis_time - Total column-organized synopsis table time monitor element |
| total_col_time | xs:nonNegativeInteger | total_col_time - Total column-organized time |
| total_col_vector_consumers | xs:long | total_col_vector_consumers - Total columnar_vector consumers |
| total_commit_proc_time | xs:nonNegativeInteger | total_commit_proc_time - Total commits processing time |
| total_commit_time | xs:nonNegativeInteger | total_commit_time - Total commit time |
| total_compilations | xs:nonNegativeInteger | total_compilations - Total compilations |
| total_compile_proc_time | xs:nonNegativeInteger | total_compile_proc_time - Total compile processing time |
| total_compile_time | xs:nonNegativeInteger | total_compile_time - Total compile time |
| total_connect_authentications | xs:nonNegativeInteger | total_connect_authentications - Connections or switch user authentications performed |
| total_connect_authentication_proc_time | xs:nonNegativeInteger | total_connect_authentication_proc_time - Total connection authentication processing time |
| total_connect_authentication_time | xs:nonNegativeInteger | total_connect_authentication_time - Total connection or switch user authentication request time |
| total_connect_request_proc_time | xs:nonNegativeInteger | total_connect_request_proc_time - Total connection or switch user request processing time |
| total_connect_request_time | xs:nonNegativeInteger | total_connect_request_time - Total connection or switch user request time |
| total_connect_requests | xs:nonNegativeInteger | total_connect_requests - Connection or switch user requests |
| total_cpu_time | xs:nonNegativeInteger | total_cpu_time - Total CPU time |
| total_disp_run_queue_time | xs:long | total_disp_run_queue_time - Total dispatcher run queue time |
| total_extended_latch_wait_time | xs:nonNegativeInteger | total_extended_latch_wait_time - Total extended latch wait time |
| total_extended_latch_waits | xs:nonNegativeInteger | total_extended_latch_waits - Total extended latch waits |
| total_hash_grpbys | xs:long | total_hash_grpbys - Total hash group by operations |
| total_hash_joins | xs:long | total_hash_joins - Total Hash Joins monitor element |
| total_hash_loops | xs:long | total_hash_loops - Total Hash Loops monitor element |
| total_implicit_compilations | xs:nonNegativeInteger | total_implicit_compilations - Total implicit complications |
| total_implicit_compile_time | xs:nonNegativeInteger | total_implicit_compile_time - Total implicit compile time |
| total_implicit_compile_proc_time | xs:nonNegativeInteger | total_implicit_compile_proc_time - Total implicit compile processing time |
| total_index_build_proc_time | xs:long | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation, recreation or rebuilding |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|-----------------------------------|-----------------------|--|
| total_index_build_time | xs:long | total_index_build_time - Total time spent building indexes due to index creation, recreation or rebuilding |
| total_indexes_built | xs:long | total_indexes_built - Total number of indexes built |
| total_load_proc_time | xs:nonNegativeInteger | total_load_proc_time - Total load processing time |
| total_load_time | xs:nonNegativeInteger | total_load_time - Total load time |
| total_loads | xs:nonNegativeInteger | total_loads - Total loads |
| total_olap_funcs | xs:long | total_olap_funcs - Total OLAP Functions monitor element |
| total_peas | xs:long | total_peas - Total partial early aggregations |
| total_peds | xs:long | total_peds - Total partial early distincts |
| total_reorg_proc_time | xs:nonNegativeInteger | total_reorg_proc_time - Total reorganization processing time |
| total_reorg_time | xs:nonNegativeInteger | total_reorg_time - Total reorganization time |
| total_reorgs | xs:nonNegativeInteger | total_reorgs - Total reorganizations |
| total_rollback_proc_time | xs:nonNegativeInteger | total_rollback_proc_time - Total rollback processing time |
| total_rollback_time | xs:nonNegativeInteger | total_rollback_time - Total rollback time |
| total_routine_invocations | xs:nonNegativeInteger | total_routine_invocations - Total routine invocations |
| total_routine_time | xs:nonNegativeInteger | total_routine_time - Total routine time |
| total_routine_user_code_time | xs:nonNegativeInteger | total_routine_user_code_time - Total routine user code time |
| total_routine_user_code_proc_time | xs:nonNegativeInteger | total_routine_user_code_proc_time - Total routine user code processing time |
| total_rqst_time | xs:nonNegativeInteger | total_rqst_time - Total request time |
| total_runstats | xs:nonNegativeInteger | total_runstats - Total runtime statistics |
| total_runstats_proc_time | xs:nonNegativeInteger | total_runstats_proc_time - Total runtime statistics processing time |
| total_runstats_time | xs:nonNegativeInteger | total_runstats_time - Total runtime statistics |
| total_section_proc_time | xs:nonNegativeInteger | total_section_proc_time - Total section processing time |
| total_section_sort_proc_time | xs:nonNegativeInteger | total_section_sort_proc_time - Total section sort processing time |
| total_section_sort_time | xs:nonNegativeInteger | total_section_sort_time - Total section sort time |
| total_section_sorts | xs:nonNegativeInteger | total_section_sorts - Total section sorts |
| total_section_time | xs:nonNegativeInteger | total_section_time - Total section time |
| total_sorts | xs:nonNegativeInteger | total_sorts - Total Sorts |
| total_stats_fabrication_proc_time | xs:nonNegativeInteger | total_stats_fabrication_proc_time - Total statistics fabrication processing time |
| total_stats_fabrication_time | xs:nonNegativeInteger | total_stats_fabrication_time - Total statistics fabrication time |
| total_stats_fabrications | xs:nonNegativeInteger | total_stats_fabrications - Total statistics fabrications |
| total_sync_runstats | xs:nonNegativeInteger | total_sync_runstats - Total synchronous RUNSTATS activities |

Table 170. Detailed metrics returned for MON_GET_UNIT_OF_WORK_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|-------------------------------|-----------------------|--|
| total_sync_runstats_proc_time | xs:nonNegativeInteger | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time |
| total_sync_runstats_time | xs:nonNegativeInteger | total_sync_runstats_time - Total synchronous RUNSTATS time |
| total_wait_time | xs:nonNegativeInteger | total_wait_time - Total wait time |
| tq_sort_heap_rejections | xs:long | tq_sort_heap_rejections - Table queue sort heap rejections |
| tq_sort_heap_requests | xs:long | tq_sort_heap_requests - Table queue sort heap requests |
| tq_tot_send_spills | xs:nonNegativeInteger | tq_tot_send_spills - Total number of table queue buffers overflowed |
| uid_sql_stmts | xs:long | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| uow_comp_status | xs:string | uow_comp_status - Unit of Work Completion Status monitor element |
| uow_id | xs:nonNegativeInteger | uow_id - Unit of work ID |
| uow_log_space_used | xs:nonNegativeInteger | uow_log_space_used - Unit of Work Log Space Used |
| uow_start_time | xs:dateTime | uow_start_time - Unit of Work Start Timestamp |
| uow_stop_time | xs:dateTime | uow_stop_time - Unit of work stop timestamp monitor element |
| wlm_queue_assignments_total | xs:nonNegativeInteger | wlm_queue_assignments_total - Workload manager total queue assignments |
| wlm_queue_time_total | xs:nonNegativeInteger | wlm_queue_time_total - Workload manager total queue time |
| workload_name | xs:string (128) | workload_name - Workload name |
| workload_occurrence_id | xs:nonNegativeInteger | workload_occurrence_id - Workload occurrence identifier This ID does not uniquely identify the workload occurrence unless it is coupled with the coordinator member and the workload name. |
| workload_occurrence_state | xs:string | workload_occurrence_state - Workload occurrence state |
| xquery_stmts | xs:long | xquery_stmts - XQuery Statements Attempted monitor element |

Usage notes

The metrics returned by the MON_GET_UNIT_OF_WORK_DETAILS function represent the accumulation of all metrics for requests that were submitted during a unit of work. This function is similar to the MON_GET_UNIT_OF_WORK table function:

- The MON_GET_UNIT_OF_WORK table function returns the most commonly used metrics in a column based format and is the most performance efficient method of retrieving metrics.
- The MON_GET_UNIT_OF_WORK_DETAILS table function returns the entire set of available metrics in an XML document format, which provides maximum

flexibility for formatting output. The XML based output can be parsed directly by an XML parser, or it can be converted to relational format by the XMLTABLE function (see the example).

Metrics are rolled up periodically during the unit of work. Therefore, the values reported by this table function reflect the current state of the system at the time of the most recent rollup. Metrics are strictly increasing in value. To determine the value of a given metric for an interval of time, use the MON_GET_UNIT_OF_WORK_DETAILS table function to query the metric at the start and end of the interval, and compute the difference.

Request metrics are controlled through the COLLECT REQUEST METRICS clause on service superclasses and the **mon_req_metrics** database configuration parameter at the database level. Metrics are only collected for a request if the request is processed by an agent in a service subclass whose parent service superclass has request metrics enabled, or if request metrics collection is enabled for the entire database. By default request metrics are enabled at the database level. If request metrics have been disabled at the database level, and for a service superclass, the metrics reported for each unit of work that are mapped to that service superclass stop increasing (or remain at 0 if request metrics were disabled at database activation time).

The MON_GET_UNIT_OF_WORK_DETAILS table function returns one row of data per unit of work and per member. No aggregation across units of work (on a member), or across members (for a service class or more), is performed. However, aggregation can be achieved through SQL queries. The input parameters have the effect of being ANDed together.

The schema for the XML document that is returned in the DETAILS column is available in the file sqllib/misc/DB2MonRoutines.xsd. Further details can be found in the file sqllib/misc/DB2MonCommon.xsd.

Example

Identify the units of work that are consuming the highest amount of CPU time on the system.

```
SELECT detmetrics.application_handle,
       detmetrics.uow_id,
       detmetrics.total_cpu_time,
       detmetrics.app_rqsts_completed_total,
       detmetrics.rqsts_completed_total
FROM TABLE(MON_GET_UNIT_OF_WORK_DETAILS(NULL,-2)) AS UOWMETRICS,
XMLTABLE (
  XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'),
  '$detmetric/db2_unit_of_work' PASSING
  XMLPARSE(DOCUMENT UOWMETRICS.DETAILS)
  as "detmetric"
) AS DETMETRICS
COLUMNS
  "APPLICATION_HANDLE" INTEGER PATH 'application_handle',
  "UOW_ID" INTEGER PATH 'uow_id',
  "TOTAL_CPU_TIME" INTEGER PATH 'system_metrics/total_cpu_time',
  "APP_RQSTS_COMPLETED_TOTAL" INTEGER
  PATH 'system_metrics/app_rqsts_completed_total',
  "RQSTS_COMPLETED_TOTAL" INTEGER
  PATH 'system_metrics/rqsts_completed_total'
) AS DETMETRICS
ORDER BY total_cpu_time DESC
```

The following is an example of output from this query.

| APPLICATION_HANDLE | UOW_ID | TOTAL_CPU_TIME | ... |
|--------------------|--------|----------------|-----|
| ----- | ----- | ----- | --- |
| 46 | 5 | 27959 | ... |

1 record(s) selected.

Output for query (continued).

| ... | APP_RQSTS_COMPLETED_TOTAL | RQSTS_COMPLETED_TOTAL |
|-----|---------------------------|-----------------------|
| --- | --- | --- |
| ... | 72 | 48 |

MON_GET_USAGE_LIST_STATUS table function - Returns the status on a usage list

The MON_GET_USAGE_LIST_STATUS table function returns current status of a usage list.

Syntax

```

►► MON_GET_USAGE_LIST_STATUS (—usagelistschema—, —————►
►—usagelistname—, —member—) —————►◄

```

The schema is SYSPROC.

Table function parameters

usagelistschema

An input argument of type VARCHAR(128) that specifies a valid schema name in the currently connected database when calling this function. If the argument is NULL or an empty string, usage lists are retrieved in all schemas in the database. If the argument is specified, usage lists are only returned for the specified schema.

usagelistname

An input argument of type VARCHAR(128) that specifies a usage list that resides in the currently connected database when calling this function. If *usagelistname* is NULL or an empty string, then the status for all usage lists from the schemas identified by the *usagelistschema* that exist are retrieved. If specified, only the status for the usage list specified from the schemas identified by the *usagelistschema* is returned.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Usage notes

The MON_GET_USAGE_LIST_STATUS function does not report information about usage lists that are in the released state. A usage list is considered to be in the released state if it is defined and has not been activated (explicitly or automatically) or has been released using the SET USAGE LIST STATE statement.

If a usage list is activated (explicitly or automatically) then the state of the usage list is set to activation pending and the memory allocation for the usage list is deferred until the table or index for which the usage list is defined is first referenced by a section. At which point the state of the usage list is set to active. If the memory for the usage list cannot be allocated, then the state of the usage list is set to failed and it must be explicitly activated using the SET USAGE LIST STATE statement.

Information returned

Table 171. Information returned for MON_GET_USAGE_LIST_STATUS

| Column Name | Data Type | Description |
|-------------------|--------------|---|
| USAGELISTSHEMA | VARCHAR(128) | usage_list_schema - Usage list schema |
| USAGELISTNAME | VARCHAR(128) | usage_list_name - Usage list name |
| OBJECTSCHEMA | VARCHAR(128) | object_schema - Object schema |
| OBJECTNAME | VARCHAR(128) | object_name - Object name |
| OBJECTTYPE | CHAR(1) | Type of object. One of: <ul style="list-style-type: none">• I - For Index• T - For Table |
| MEMBER | SMALLINT | member - Database member |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data partition identifier |
| STATE | CHAR(1) | usage_list_state - Usage list state |
| LAST_STATE_CHANGE | TIMESTAMP | usage_list_last_state_change - Last state change |
| LIST_SIZE | INTEGER | usage_list_size - Usage list size |
| USED_ENTRIES | INTEGER | usage_list_used_entries - Usage list used entries |
| LIST_MEM_SIZE | INTEGER | usage_list_mem_size - Usage list memory size |
| WRAPPED | CHAR(1) | usage_list_wrapped - Usage list wrap indicator |

MON_GET_UTILITY table function - Get utilities running on the database

The MON_GET_UTILITY table function returns information about utilities currently running on the database.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority

- SQLADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_GET_UTILITY—(—*member*—)——►►

The schema is SYSPROC.

Routine parameters

member

An input parameter of type INTEGER that specifies a valid database member from which the records should be fetched. Specify -1 or null for the current member, or -2 for information from all active database members. An active database member is a database that is available for connection and use by applications.

Information returned

Table 172. Information returned by the MON_GET_UTILITY routine

| Column name | Data type | Description |
|-----------------------|--------------------------|---|
| MEMBER | SMALLINT | member - Database member monitor element |
| COORD_MEMBER | SMALLINT | coord_member - Coordinator member monitor element |
| APPLICATION_HANDLE | BIGINT | application_handle - Application handle monitor element |
| APPLICATION_ID | VARCHAR(128) | appl_id - Application ID monitor element |
| APPLICATION_NAME | VARCHAR(128) | appl_name - Application name monitor element |
| SESSION_AUTH_ID | VARCHAR(128) | session_auth_id - Session authorization ID monitor element |
| CLIENT_WRKSTNNAME | VARCHAR(255) | client_wrkstnname - Client workstation name monitor element |
| CLIENT_ACCTNG | VARCHAR(255) | client_acctng - Client accounting string monitor element |
| CLIENT_USERID | VARCHAR(255) | client_userid - Client user ID monitor element |
| CLIENT_APPLNAME | VARCHAR(255) | client_applname - Client application name monitor element |
| UTILITY_INVOCATION_ID | VARCHAR(32) FOR BIT DATA | utility_invocation_id - Utility invocation ID monitor element |
| UTILITY_ID | INTEGER | utility_id - Utility ID
Unique to a database partition. |
| UTILITY_START_TIME | TIMESTAMP | utility_start_time - Utility start time |
| UTILITY_TYPE | VARCHAR(16) | utility_type - Utility type |

Table 172. Information returned by the MON_GET_UTILITY routine (continued)

| Column name | Data type | Description |
|------------------------|--------------|--|
| UTILITY_OPERATION_TYPE | CHAR(1) | utility_operation_type - Utility operation type monitor element |
| UTILITY_INVOKER_TYPE | VARCHAR(4) | utility_invoker_type - Utility invoker type |
| UTILITY_PRIORITY | INTEGER | utility_priority - Utility priority

Priority if utility supports throttling, otherwise null. |
| OBJECT_TYPE | VARCHAR(16) | Type of object. One of: <ul style="list-style-type: none"> • DATABASE • INDEX • PARTITIONGROUP • TABLE • TABLESPACE • VIEW |
| OBJECT_SCHEMA | VARCHAR(128) | object_schema - Object schema monitor element |
| OBJECT_NAME | VARCHAR(128) | object_name - Object name monitor element |
| NUM_TBSPS | INTEGER | num_tbsps - Number of table spaces monitor element |
| TBSP_NAMES | CLOB(5M) | tbsp_names - Table space names monitor element |
| UTILITY_DETAIL | CLOB(2M) | utility_detail - Utility detail monitor element |

Example

List all LOAD commands currently running on the database server and provide information about the applications and the users running the command.

```
SELECT COORD_MEMBER, APPLICATION_HANDLE AS APPHDL,
       SUBSTR(APPLICATION_NAME, 1, 10) AS APPNAME,
       SUBSTR(SESSION_AUTH_ID, 1, 10) AS USER,
       SUBSTR(UTILITY_DETAIL, 1, 150) AS CMD
FROM TABLE(MON_GET_UTILITY(-2)) AS T
```

The following result is output:

```
COORD_MEMBER APPHDL          APPNAME    USER      ...
-----
          0          736 db2bp    USER2     ...
          0          724 db2bp    USER1     ...

... CMD
...
... [LOADID: 63.2013-02-01-09.48.42.976823.0 (2;21)] [*LOCAL.user2.130201144359] ...
... [LOADID: 20.2013-02-01-09.43.49.510806.0 (2;20)] [*LOCAL.user1.130201144240] ...

... CMD (continued)
...
... OFFLINE LOAD DEL AUTOMATIC INDEXING INSERT COPY NO SCHEMA2.T2
... OFFLINE LOAD DEL AUTOMATIC INDEXING INSERT COPY NO SCHEMA1.T1
```

MON_GET_WORKLOAD table function - Get workload metrics

The MON_GET_WORKLOAD table function returns metrics for one or more workloads.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_GET_WORKLOAD—(—*workload_name*—,—*member*—)————►►

The schema is SYSPROC.

Routine parameters

workload_name

An input argument of type VARCHAR(128) that specifies a specific workload for which the metrics are to be returned. If the argument is NULL or an empty string, metrics are returned for all workloads.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is set implicitly.

Information returned

Table 173. Information returned for MON_GET_WORKLOAD

| Column Name | Data Type | Description or corresponding monitor element |
|-------------------------|--------------|---|
| WORKLOAD_NAME | VARCHAR(128) | workload_name - Workload name |
| WORKLOAD_ID | INTEGER | workload_id - Workload ID |
| MEMBER | SMALLINT | member - Database member |
| ACT_ABORTED_TOTAL | BIGINT | act_aborted_total - Total aborted activities |
| ACT_COMPLETED_TOTAL | BIGINT | act_completed_total - Total completed activities |
| ACT_REJECTED_TOTAL | BIGINT | act_rejected_total - Total rejected activities |
| AGENT_WAIT_TIME | BIGINT | agent_wait_time - Agent wait time |
| AGENT_WAITS_TOTAL | BIGINT | agent_waits_total - Total agent waits |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer pool XDA data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer pool XDA data physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer pool XDA data writes |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| CLIENT_IDLE_WAIT_TIME | BIGINT | client_idle_wait_time - Client idle wait time |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| FCM_RECV_VOLUME | BIGINT | fcm_recv_volume - FCM recv volume |
| FCM_RECVS_TOTAL | BIGINT | fcm_recvs_total - FCM recvs total |
| FCM_SEND_VOLUME | BIGINT | fcm_send_volume - FCM send volume |
| FCM_SENDS_TOTAL | BIGINT | fcm_sends_total - FCM sends total |
| FCM_RECV_WAIT_TIME | BIGINT | fcm_recv_wait_time - FCM recv wait time |
| FCM_SEND_WAIT_TIME | BIGINT | fcm_send_wait_time - FCM send wait time |
| IPC_RECV_VOLUME | BIGINT | ipc_recv_volume - Interprocess communication recv volume |
| IPC_RECV_WAIT_TIME | BIGINT | ipc_recv_wait_time - Interprocess communication recv wait time |
| IPC_RECVS_TOTAL | BIGINT | ipc_recvs_total - Interprocess communication recvs total |
| IPC_SEND_VOLUME | BIGINT | ipc_send_volume - Interprocess communication send volume |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|------------------------------|-----------|--|
| IPC_SEND_WAIT_TIME | BIGINT | ipc_send_wait_time - Interprocess communication send wait time |
| IPC_SENDS_TOTAL | BIGINT | ipc_sends_total - Interprocess communication send total |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOG_BUFFER_WAIT_TIME | BIGINT | log_buffer_wait_time - Log buffer wait time |
| NUM_LOG_BUFFER_FULL | BIGINT | num_log_buffer_full - Number of full log buffers |
| LOG_DISK_WAIT_TIME | BIGINT | log_disk_wait_time - Log disk wait time |
| LOG_DISK_WAITS_TOTAL | BIGINT | log_disk_waits_total - Log disk waits total |
| RQSTS_COMPLETED_TOTAL | BIGINT | rqsts_completed_total - Total requests completed |
| ROWS_MODIFIED | BIGINT | rows_modified - Rows modified |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_RETURNED | BIGINT | rows_returned - Rows returned |
| TCPIP_RECV_VOLUME | BIGINT | tcipip_recv_volume - TCP/IP received volume |
| TCPIP_SEND_VOLUME | BIGINT | tcipip_send_volume - TCP/IP send volume |
| TCPIP_RECV_WAIT_TIME | BIGINT | tcipip_recv_wait_time - TCP/IP recv wait time |
| TCPIP_RECVS_TOTAL | BIGINT | tcipip_recv_total - TCP/IP recvs total |
| TCPIP_SEND_WAIT_TIME | BIGINT | tcipip_send_wait_time - TCP/IP send wait time |
| TCPIP_SENDS_TOTAL | BIGINT | tcipip_sends_total - TCP/IP sends total |
| TOTAL_APP_RQST_TIME | BIGINT | total_app_rqst_time - Total application request time |
| TOTAL_RQST_TIME | BIGINT | total_rqst_time - Total request time |
| WLM_QUEUE_TIME_TOTAL | BIGINT | wlm_queue_time_total - Workload manager total queue time |
| WLM_QUEUE_ASSIGNMENTS_TOTAL | BIGINT | wlm_queue_assignments_total - Workload manager total queue assignments |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time |
| TOTAL_WAIT_TIME | BIGINT | total_wait_time - Total wait time |
| APP_RQSTS_COMPLETED_TOTAL | BIGINT | app_rqsts_completed_total - Total application requests completed |
| TOTAL_SECTION_SORT_TIME | BIGINT | total_section_sort_time - Total section sort time |
| TOTAL_SECTION_SORT_PROC_TIME | BIGINT | total_section_sort_proc_time - Total section sort processing time |
| TOTAL_SECTION_SORTS | BIGINT | total_section_sorts - Total section sorts |
| TOTAL_SORTS | BIGINT | total_sorts - Total Sorts |
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post threshold sorts |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post shared threshold sorts |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| TOTAL_COMPILE_TIME | BIGINT | total_compile_time - Total compile time |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|----------------------------------|-----------|---|
| TOTAL_COMPILE_PROC_TIME | BIGINT | total_compile_proc_time - Total compile processing time |
| TOTAL_COMPILATIONS | BIGINT | total_compilations - Total compilations |
| TOTAL_IMPLICIT_COMPILE_TIME | BIGINT | total_implicit_compile_time - Total implicit compile time |
| TOTAL_IMPLICIT_COMPILE_PROC_TIME | BIGINT | total_implicit_compile_proc_time - Total implicit compile processing time |
| TOTAL_IMPLICIT_COMPILATIONS | BIGINT | total_implicit_compilations - Total implicit complications |
| TOTAL_SECTION_TIME | BIGINT | total_section_time - Total section time |
| TOTAL_SECTION_PROC_TIME | BIGINT | total_section_proc_time - Total section processing time |
| TOTAL_APP_SECTION_EXECUTIONS | BIGINT | total_app_section_executions - Total section executions |
| TOTAL_ACT_TIME | BIGINT | total_act_time - Total activity time monitor element |
| TOTAL_ACT_WAIT_TIME | BIGINT | total_act_wait_time - Total activity wait time monitor element |
| ACT_RQSTS_TOTAL | BIGINT | act_rqsts_total - Total activity requests |
| TOTAL_ROUTINE_TIME | BIGINT | total_routine_time - Total routine time |
| TOTAL_ROUTINE_INVOCATIONS | BIGINT | total_routine_invocations - Total routine invocations |
| TOTAL_COMMIT_TIME | BIGINT | total_commit_time - Total commit time |
| TOTAL_COMMIT_PROC_TIME | BIGINT | total_commit_proc_time - Total commits processing time |
| TOTAL_APP_COMMITS | BIGINT | total_app_commits - Total application commits |
| INT_COMMITS | BIGINT | int_commits - Internal commits |
| TOTAL_ROLLBACK_TIME | BIGINT | total_rollback_time - Total rollback time |
| TOTAL_ROLLBACK_PROC_TIME | BIGINT | total_rollback_proc_time - Total rollback processing time |
| TOTAL_APP_ROLLBACKS | BIGINT | total_app_rollbacks - Total application rollbacks |
| INT_ROLLBACKS | BIGINT | int_rollbacks - Internal rollbacks |
| TOTAL_RUNSTATS_TIME | BIGINT | total_runstats_time - Total runtime statistics |
| TOTAL_RUNSTATS_PROC_TIME | BIGINT | total_runstats_proc_time - Total runtime statistics processing time |
| TOTAL_RUNSTATS | BIGINT | total_runstats - Total runtime statistics |
| TOTAL_REORG_TIME | BIGINT | total_reorg_time - Total reorganization time |
| TOTAL_REORG_PROC_TIME | BIGINT | total_reorg_proc_time - Total reorganization processing time |
| TOTAL_REORGS | BIGINT | total_reorgs - Total reorganizations |
| TOTAL_LOAD_TIME | BIGINT | total_load_time - Total load time |
| TOTAL_LOAD_PROC_TIME | BIGINT | total_load_proc_time - Total load processing time |
| TOTAL_LOADS | BIGINT | total_loads - Total loads |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|--------------------------------|-----------|--|
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups |
| THRESH_VIOLATIONS | BIGINT | hresh_violations - Number of threshold violations |
| NUM_LW_THRESH_EXCEEDED | BIGINT | num_lw_thresh_exceeded - Number of thresholds exceeded |
| LOCK_WAITS_GLOBAL | BIGINT | lock_waits_global - Lock waits global |
| LOCK_WAIT_TIME_GLOBAL | BIGINT | lock_wait_time_global - Lock wait time global |
| LOCK_TIMEOUTS_GLOBAL | BIGINT | lock_timeouts_global - Lock timeouts global |
| LOCK_ESCALS_MAXLOCKS | BIGINT | lock_escals_maxlocks - Number of maxlocks lock escalations |
| LOCK_ESCALS_LOCKLIST | BIGINT | lock_escals_locklist - Number of locklist lock escalations |
| LOCK_ESCALS_GLOBAL | BIGINT | lock_escals_global - Number of global lock escalations |
| RECLAIM_WAIT_TIME | BIGINT | reclaim_wait_time - Reclaim wait time |
| SPACEMAPPAGE_RECLAIM_WAIT_TIME | BIGINT | spacemappage_reclaim_wait_time - Space map page reclaim wait time |
| CF_WAITS | BIGINT | cf_waits - Number of cluster caching facility waits |
| CF_WAIT_TIME | BIGINT | cf_wait_time - cluster caching facility wait time |
| POOL_DATA_GBP_L_READS | BIGINT | pool_data_gbp_l_reads - Group buffer pool data logical reads |
| POOL_DATA_GBP_P_READS | BIGINT | pool_data_gbp_p_reads - Group buffer pool data physical reads |
| POOL_DATA_LBP_PAGES_FOUND | BIGINT | pool_data_lbp_pages_found - Local buffer pool found data pages |
| POOL_DATA_GBP_INVALID_PAGES | BIGINT | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages |
| POOL_INDEX_GBP_L_READS | BIGINT | pool_index_gbp_l_reads - Group buffer pool index logical reads |
| POOL_INDEX_GBP_P_READS | BIGINT | pool_index_gbp_p_reads - Group buffer pool index physical reads |
| POOL_INDEX_LBP_PAGES_FOUND | BIGINT | pool_index_lbp_pages_found - Local buffer pool index pages found |
| POOL_INDEX_GBP_INVALID_PAGES | BIGINT | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages |
| POOL_XDA_GBP_L_READS | BIGINT | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |
| POOL_XDA_GBP_P_READS | BIGINT | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |
| POOL_XDA_LBP_PAGES_FOUND | BIGINT | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |
| POOL_XDA_GBP_INVALID_PAGES | BIGINT | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| AUDIT_EVENTS_TOTAL | BIGINT | audit_events_total - Total audit events |
| AUDIT_FILE_WRITES_TOTAL | BIGINT | audit_file_writes_total - Total Audit files written |
| AUDIT_FILE_WRITE_WAIT_TIME | BIGINT | audit_file_write_wait_time - Audit file write wait time |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|-----------------------------------|-----------|--|
| AUDIT_SUBSYSTEM_WAITS_TOTAL | BIGINT | audit_subsystem_waits_total - Total audit subsystem waits |
| AUDIT_SUBSYSTEM_WAIT_TIME | BIGINT | audit_subsystem_wait_time - Audit subsystem wait time |
| DIAGLOG_WRITES_TOTAL | BIGINT | diaglog_writes_total - Diag log total writes |
| DIAGLOG_WRITE_WAIT_TIME | BIGINT | diaglog_write_wait_time - Diag log write time |
| FCM_MESSAGE_RECVS_TOTAL | BIGINT | fcm_message_recvs_total - FCM message recvs total |
| FCM_MESSAGE_RECV_VOLUME | BIGINT | fcm_message_recv_volume - FCM message recv volume |
| FCM_MESSAGE_RECV_WAIT_TIME | BIGINT | fcm_message_recv_wait_time - FCM message recv wait time |
| FCM_MESSAGE_SENDS_TOTAL | BIGINT | fcm_message_sends_total - FCM message sends total |
| FCM_MESSAGE_SEND_VOLUME | BIGINT | fcm_message_send_volume - FCM message send volume |
| FCM_MESSAGE_SEND_WAIT_TIME | BIGINT | fcm_message_send_wait_time - FCM message send wait time |
| FCM_TQ_RECVS_TOTAL | BIGINT | fcm_tq_recvs_total - FCM tablequeue recvs total |
| FCM_TQ_RECV_VOLUME | BIGINT | fcm_tq_recv_volume - FCM tablequeue recv volume |
| FCM_TQ_RECV_WAIT_TIME | BIGINT | fcm_tq_recv_wait_time - FCM tablequeue recv wait time |
| FCM_TQ_SENDS_TOTAL | BIGINT | fcm_tq_sends_total - FCM tablequeue send total |
| FCM_TQ_SEND_VOLUME | BIGINT | fcm_tq_send_volume - FCM tablequeue send volume |
| FCM_TQ_SEND_WAIT_TIME | BIGINT | fcm_tq_send_wait_time - FCM tablequeue send wait time |
| TOTAL_ROUTINE_USER_CODE_PROC_TIME | BIGINT | total_routine_user_code_proc_time - Total routine user code processing time |
| TOTAL_ROUTINE_USER_CODE_TIME | BIGINT | total_routine_user_code_time - Total routine user code time |
| TQ_TOT_SEND_SPILLS | BIGINT | tq_tot_send_spills - Total number of table queue buffers overflowed |
| EVMON_WAIT_TIME | BIGINT | evmon_wait_time - Event monitor wait time |
| EVMON_WAITS_TOTAL | BIGINT | evmon_waits_total - Event monitor total waits |
| TOTAL_EXTENDED_LATCH_WAIT_TIME | BIGINT | total_extended_latch_wait_time - Total extended latch wait time |
| TOTAL_EXTENDED_LATCH_WAITS | BIGINT | total_extended_latch_waits - Total extended latch waits |
| TOTAL_STATS_FABRICATION_TIME | BIGINT | total_stats_fabrication_time - Total statistics fabrication time |
| TOTAL_STATS_FABRICATION_PROC_TIME | BIGINT | total_stats_fabrication_proc_time - Total statistics fabrication processing time |
| TOTAL_STATS_FABRICATIONS | BIGINT | total_stats_fabrications - Total statistics fabrications |
| TOTAL_SYNC_RUNSTATS_TIME | BIGINT | total_sync_runstats_time - Total synchronous RUNSTATS time |
| TOTAL_SYNC_RUNSTATS_PROC_TIME | BIGINT | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|------------------------------------|-----------|--|
| TOTAL_SYNC_RUNSTATS | BIGINT | total_sync_runstats - Total synchronous RUNSTATS activities |
| TOTAL_DISP_RUN_QUEUE_TIME | BIGINT | total_disp_run_queue_time - Total dispatcher run queue time |
| TOTAL_PEDS | BIGINT | total_peds - Total partial early distincts |
| DISABLED_PEDS | BIGINT | disabled_peds - Disabled partial early distincts |
| POST_THRESHOLD_PEDS | BIGINT | post_threshold_peds - Partial early distincts threshold |
| TOTAL_PEAS | BIGINT | total_peas - Total partial early aggregations |
| POST_THRESHOLD_PEAS | BIGINT | post_threshold_peas - Partial early aggregation threshold |
| TQ_SORT_HEAP_REQUESTS | BIGINT | tq_sort_heap_requests - Table queue sort heap requests |
| TQ_SORT_HEAP_REJECTIONS | BIGINT | tq_sort_heap_rejections - Table queue sort heap rejections |
| POOL_QUEUED_ASYNC_DATA_REQS | BIGINT | pool_queued_async_data_reqs - Data prefetch requests monitor element |
| POOL_QUEUED_ASYNC_INDEX_REQS | BIGINT | pool_queued_async_index_reqs - Index prefetch requests monitor element |
| POOL_QUEUED_ASYNC_XDA_REQS | BIGINT | pool_queued_async_xda_reqs - XDA prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_OTHER_REQS | BIGINT | pool_queued_async_other_reqs - Non-prefetch requests monitor element |
| POOL_QUEUED_ASYNC_DATA_PAGES | BIGINT | pool_queued_async_data_pages - Data pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_INDEX_PAGES | BIGINT | pool_queued_async_index_pages - Index pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_XDA_PAGES | BIGINT | pool_queued_async_xda_pages - XDA pages prefetch requests monitor element |
| POOL_QUEUED_ASYNC_TEMP_DATA_PAGES | BIGINT | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_INDEX_PAGES | BIGINT | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element |
| POOL_QUEUED_ASYNC_TEMP_XDA_PAGES | BIGINT | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_DATA_REQS | BIGINT | pool_failed_async_data_reqs - Failed data prefetch requests monitor element |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|--|
| POOL_FAILED_ASYNC_INDEX_REQS | BIGINT | pool_failed_async_index_reqs - Failed index prefetch requests monitor element |
| POOL_FAILED_ASYNC_XDA_REQS | BIGINT | pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element |
| POOL_FAILED_ASYNC_TEMP_DATA_REQS | BIGINT | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_TEMP_INDEX_REQS | BIGINT | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_TEMP_XDA_REQS | BIGINT | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element |
| POOL_FAILED_ASYNC_OTHER_REQS | BIGINT | pool_failed_async_other_reqs - Failed non-prefetch requests monitor element |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| PREFETCH_WAITS | BIGINT | prefetch_waits - Prefetcher wait count monitor element |
| APP_ACT_COMPLETED_TOTAL | BIGINT | app_act_completed_total - Total successful external coordinator activities monitor element |
| APP_ACT_ABORTED_TOTAL | BIGINT | app_act_aborted_total - Total failed external coordinator activities monitor element |
| APP_ACT_REJECTED_TOTAL | BIGINT | app_act_rejected_total - Total rejected external coordinator activities monitor element |
| TOTAL_CONNECT_REQUEST_TIME | BIGINT | total_connect_request_time - Total connection or switch user request time monitor element |
| TOTAL_CONNECT_REQUEST_PROC_TIME | BIGINT | total_connect_request_proc_time - Total connection or switch user request processing time monitor element |
| TOTAL_CONNECT_REQUESTS | BIGINT | total_connect_requests - Connection or switch user requests monitor element |
| TOTAL_CONNECT_AUTHENTICATION_TIME | BIGINT | total_connect_authentication_time - Total connection or switch user authentication request time monitor element |
| TOTAL_CONNECT_AUTHENTICATION_PROC_TIME | BIGINT | total_connect_authentication_proc_time - Total connection authentication processing time monitor element |
| TOTAL_CONNECT_AUTHENTIFICATIONS | BIGINT | total_connect_authentications - Connections or switch user authentications performed monitor element |
| POOL_DATA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| POOL_INDEX_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| POOL_XDA_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|------------------------------|-----------|---|
| COMM_EXIT_WAIT_TIME | BIGINT | comm_exit_wait_time - Communication buffer exit wait time monitor element |
| COMM_EXIT_WAITS | BIGINT | comm_exit_waits - Communication buffer exit number of waits monitor element |
| IDA_SEND_WAIT_TIME | BIGINT | ida_send_wait_time - Time spent waiting to send data monitor element |
| IDA_SENDS_TOTAL | BIGINT | ida_sends_total - Number of times data sent monitor element |
| IDA_SEND_VOLUME | BIGINT | ida_send_volume - Total data volume sent monitor element |
| IDA_RECV_WAIT_TIME | BIGINT | ida_rcv_wait_time - Time spent waiting to receive data monitor element |
| IDA_RECVS_TOTAL | BIGINT | ida_rcvs_total - Number of times data received monitor element |
| IDA_RECV_VOLUME | BIGINT | ida_rcv_volume - Total data volume received monitor element |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted monitor element |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted monitor element |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated monitor element |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total Hash Joins monitor element |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total Hash Loops monitor element |
| HASH_JOIN_OVERFLOWS | BIGINT | hash_join_overflows - Hash Join Overflows monitor element |
| HASH_JOIN_SMALL_OVERFLOWS | BIGINT | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| POST_SHRTHRESHOLD_HASH_JOINS | BIGINT | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP Functions monitor element |
| OLAP_FUNC_OVERFLOWS | BIGINT | olap_func_overflows - OLAP Function Overflows monitor element |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element |
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL Statements Attempted monitor element |
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed Statement Operations monitor element |
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL Statements Executed monitor element |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| MERGE_SQL_STMTS | BIGINT | merge_sql_stmts - Merge SQL statements executed |
| XQUERY_STMTS | BIGINT | xquery_stmts - XQuery Statements Attempted monitor element |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---------------------------------------|-----------|---|
| IMPLICIT_REBINDS | BIGINT | implicit_rebinds - number of implicit rebinds |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/Precompiles Attempted monitor element |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal Rows Inserted monitor element |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal Rows Updated monitor element |
| CALL_SQL_STMTS | BIGINT | call_sql_stmts - CALL SQL statements executed |
| POOL_COL_L_READS | BIGINT | pool_col_l_reads - Buffer pool column-organized logical reads |
| POOL_TEMP_COL_L_READS | BIGINT | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| POOL_COL_P_READS | BIGINT | pool_col_p_reads - Buffer pool column-organized physical reads |
| POOL_TEMP_COL_P_READS | BIGINT | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| POOL_COL_LBP_PAGES_FOUND | BIGINT | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found |
| POOL_COL_WRITES | BIGINT | pool_col_writes - Buffer pool column-organized writes |
| POOL_COL_GBP_L_READS | BIGINT | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| POOL_COL_GBP_P_READS | BIGINT | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| POOL_COL_GBP_INVALID_PAGES | BIGINT | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| POOL_COL_GBP_INDEP_PAGES_FOUND_IN_LBP | BIGINT | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |
| POOL_QUEUED_ASYNC_COL_REQS | BIGINT | pool_queued_async_col_reqs - Column-organized prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_REQS | BIGINT | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| POOL_QUEUED_ASYNC_COL_PAGES | BIGINT | pool_queued_async_col_pages - Column-organized page prefetch requests |
| POOL_QUEUED_ASYNC_TEMP_COL_PAGES | BIGINT | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| POOL_FAILED_ASYNC_COL_REQS | BIGINT | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| POOL_FAILED_ASYNC_TEMP_COL_REQS | BIGINT | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |
| TOTAL_COL_TIME | BIGINT | total_col_time - Total column-organized time |
| TOTAL_COL_PROC_TIME | BIGINT | total_col_proc_time - Total column-organized processing time |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|---|
| TOTAL_COL_EXECUTIONS | BIGINT | total_col_executions - Total column-organized executions |
| POST_THRESHOLD_HASH_JOINS | BIGINT | post_threshold_hash_joins - Hash Join Threshold monitor element |
| POOL_DATA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_L_READS | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_WRITES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGE_UPDATES | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_READ_TIME | BIGINT | Reserved for future use. |
| POOL_CACHING_TIER_PAGE_WRITE_TIME | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|---|-----------|--|
| POOL_COL_CACHING_TIER_GBP_INVALID_PAGES | BIGINT | Reserved for future use. |
| POOL_DATA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_INDEX_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_XDA_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| POOL_COL_CACHING_TIER_GBP_INDEP_PAGES_FOUND | BIGINT | Reserved for future use. |
| TOTAL_HASH_GRPBY | BIGINT | total_hash_grpbys - Total hash group by operations |
| HASH_GRPBY_OVERFLOW | BIGINT | hash_grpby_overflows - Hash group by overflows |
| POST_THRESHOLD_HASH_GRPBY | BIGINT | post_threshold_hash_grpbys - Hash group by threshold |
| POST_THRESHOLD_OLAP_FUNCS | BIGINT | post_threshold_olap_funcs - OLAP function threshold monitor element |
| POST_THRESHOLD_COL_VECTOR_CONSUMERS | BIGINT | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| TOTAL_COL_VECTOR_CONSUMERS | BIGINT | total_col_vector_consumers - Total columnar_vector consumers |
| ACTIVE_HASH_GRPBY | BIGINT | active_hash_grpbys - Active hash GROUP BY operations |
| ACTIVE_HASH_JOINS | BIGINT | active_hash_joins - Active hash joins |
| ACTIVE_OLAP_FUNCS | BIGINT | active_olap_funcs - Active OLAP Functions |
| ACTIVE_PEAS | BIGINT | active_peas - Active partial early aggregations operations |
| ACTIVE_PEDS | BIGINT | active_peds - Active partial early distinct operations |
| ACTIVE_SORT_CONSUMERS | BIGINT | active_sort_consumers - Active sort consumers |
| ACTIVE_SORTS | BIGINT | active_sorts - Active Sorts |
| ACTIVE_COL_VECTOR_CONSUMERS | BIGINT | active_col_vector_consumers - Active columnar_vector consumers |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total Sort Heap Allocated |
| SORT_SHRHEAP_ALLOCATED | BIGINT | sort_shrheap_allocated - Sort Share Heap Currently Allocated |
| TOTAL_BACKUP_TIME | BIGINT | total_backup_time - Total elapsed time for online backups |
| TOTAL_BACKUP_PROC_TIME | BIGINT | total_backup_proc_time - Total non-wait time for online backups |
| TOTAL_BACKUPS | BIGINT | total_backups - Total backups |
| TOTAL_INDEX_BUILD_TIME | BIGINT | total_index_build_time - Total time spent building indexes due to index creation, recreation or rebuilding |
| TOTAL_INDEX_BUILD_PROC_TIME | BIGINT | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation, recreation or rebuilding |
| TOTAL_INDEXES_BUILT | BIGINT | total_indexes_built - Total number of indexes built |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|--------------------------------|-----------|--|
| FCM_TQ_RECV_WAITS_TOTAL | BIGINT | fcm_tq_recv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| FCM_MESSAGE_RECV_WAITS_TOTAL | BIGINT | fcm_message_recv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| FCM_TQ_SEND_WAITS_TOTAL | BIGINT | fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |
| FCM_MESSAGE_SEND_WAITS_TOTAL | BIGINT | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| FCM_SEND_WAITS_TOTAL | BIGINT | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| FCM_RECV_WAITS_TOTAL | BIGINT | fcm_recv_waits_total - Number of times spent waiting to receive data through FCM monitor element |
| EXT_TABLE_RECV_WAIT_TIME | BIGINT | ext_table_recv_wait_time - Total agent wait time for external table readers monitor element |
| EXT_TABLE_RECVS_TOTAL | BIGINT | ext_table_recvs_total - Total row batches received from external table readers monitor element |
| EXT_TABLE_RECV_VOLUME | BIGINT | ext_table_recv_volume - Total data received from external table readers monitor element |
| EXT_TABLE_READ_VOLUME | BIGINT | ext_table_read_volume - Total data read by external table readers monitor element |
| EXT_TABLE_SEND_WAIT_TIME | BIGINT | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| EXT_TABLE_SENDS_TOTAL | BIGINT | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| EXT_TABLE_SEND_VOLUME | BIGINT | ext_table_send_volume - Total data sent to external table writers monitor element |
| EXT_TABLE_WRITE_VOLUME | BIGINT | ext_table_write_volume - Total data written by external table writers monitor element |
| COL_VECTOR_CONSUMER_OVERFLOW\$ | BIGINT | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |
| TOTAL_COL_SYNOPSIS_TIME | BIGINT | total_col_synopsis_time - Total column-organized synopsis table time monitor element |
| TOTAL_COL_SYNOPSIS_PROC_TIME | BIGINT | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| TOTAL_COL_SYNOPSIS_EXECUTIONS | BIGINT | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |
| COL_SYNOPSIS_ROWS_INSERTED | BIGINT | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| LOB_PREFETCH_WAIT_TIME | BIGINT | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| LOB_PREFETCH_REQS | BIGINT | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| FED_ROWS_DELETED | BIGINT | fed_rows_deleted - Rows deleted by a federation system monitor element |

Table 173. Information returned for MON_GET_WORKLOAD (continued)

| Column Name | Data Type | Description or corresponding monitor element |
|----------------------|-----------|---|
| FED_ROWS_INSERTED | BIGINT | fed_rows_inserted - Rows inserted by a federation system monitor element |
| FED_ROWS_UPDATED | BIGINT | fed_rows_updated - Rows updated by a federation system monitor element |
| FED_ROWS_READ | BIGINT | fed_rows_read - Rows read by a federation system monitor element |
| FED_WAIT_TIME | BIGINT | fed_wait_time - Time spent by a federation server monitor element |
| FED_WAITS_TOTAL | BIGINT | fed_waits_total - Total number of execution times for a federation server monitor element |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Insert count for SQL workspace monitor element |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section lookups - SQL workspace access count monitor element |
| ADM_OVERFLOWS | BIGINT | Reserved for future use. |
| ADM_BYPASS_ACT_TOTAL | BIGINT | Reserved for future use. |

Usage notes

The metrics returned by the MON_GET_WORKLOAD table function represent the accumulation of all metrics for requests that were submitted by connections mapped to the identified workload object. Metrics are rolled up to a workload on unit of work boundaries, and periodically during the execution of requests. Therefore, the values reported by this table function reflect the current state of the system at the time of the most recent rollup. Metrics are strictly increasing in value. To determine the value of a given metric for an interval of time, use the MON_GET_WORKLOAD table function to query the metric at the start and end of the interval, and compute the difference.

Request metrics are controlled through the COLLECT REQUEST METRICS clause on service superclasses and the *mon_req_metrics* database configuration parameter at the database level. Metrics are only collected for a request if the request is processed by an agent in a service subclass whose parent service superclass has request metrics enabled, or if request metrics collection is enabled for the entire database. By default, request metrics are enabled at the database level. If request metrics have been disabled at the database level, and for a service superclass, then the metrics reported for each workload that is mapped to that service superclass stop increasing (or remain at 0 if request metrics were disabled at database activation time).

The MON_GET_WORKLOAD table function returns one row of data per workload and per member. No aggregation across workloads (on a member), or across members (for a service class or more), is performed. However, aggregation can be achieved through SQL queries (see the example).

Note that the metrics from background EDUs such as prefetchers and page cleaners are not rolled up to workloads.

Example

Display lock information for each workload, aggregated across member, ordered by highest lock wait time.

```
SELECT varchar(workload_name,30) as workload_name,
       sum(lock_wait_time) as total_lock_wait_time,
       sum(lock_waits) as total_lock_waits,
       sum(lock_timeouts) as total_lock_timeouts,
       sum(lock_escals) as total_lock_escals
FROM TABLE(MON_GET_WORKLOAD('','-2')) AS t
GROUP BY workload_name
ORDER BY total_lock_wait_time DESC
```

The following is an example of output from this query.

| WORKLOAD_NAME | TOTAL_LOCK_WAIT_TIME | TOTAL_LOCK_WAITS | ... |
|------------------------|----------------------|------------------|-----|
| SYSDEFAULTADWORKLOAD | 0 | 0 | ... |
| SYSDEFAULTUSERWORKLOAD | 0 | 0 | ... |

2 record(s) selected.

Output for query (continued).

| ... | TOTAL_LOCK_TIMEOUTS | TOTAL_LOCK_ESCALS |
|-----|---------------------|-------------------|
| ... | 0 | 0 |
| ... | 0 | 0 |

MON_GET_WORKLOAD_DETAILS table function - Get workload metrics as an XML document

The MON_GET_WORKLOAD_DETAILS table function returns metrics for one or more workloads.

The metrics are returned in an XML document.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

```
➤—MON_GET_WORKLOAD_DETAILS—(—workload_name—,—member—)—➤
```

The schema is SYSPROC.

Routine parameters

workload_name

An input argument of type VARCHAR(128) that specifies a specific workload

for which the metrics are to be returned. If the argument is NULL or an empty string, metrics are returned for all workloads.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all active database members. If the NULL value is specified, -1 is set implicitly.

Information returned

Table 174. Information returned for MON_GET_WORKLOAD_DETAILS

| Column Name | Data Type | Description |
|---------------|--------------|---|
| WORKLOAD_NAME | VARCHAR(128) | workload_name - Workload name |
| WORKLOAD_ID | INTEGER | workload_id - Workload ID |
| MEMBER | SMALLINT | member - Database member |
| DETAILS | BLOB(1M) | XML document that contains detailed metrics for the workload. See Table 175 for a description of the elements in this document. |

The following example shows the structure of the XML document that is returned in the DETAILS column.

```
<db2_workload xmlns="http://www.ibm.com/xmlns/prod/db2/mon" release="90700000">
  <workload_name>SYSDEFAULTADMWORKLOAD</workload_name>
  <workload_id>11</workload_id>
  <member>0</member>
  <system_metrics release="90700000">
    <act_aborted_total>5</act_aborted_total>
    ...
    <wlm_queue_assignments_total>3</wlm_queue_assignments_total>
  </system_metrics>
</db2_workload>
```

For the full schema, see `sqllib/misc/DB2MonRoutines.xsd`.

Table 175. Detailed metrics returned for MON_GET_WORKLOAD_DETAILS

| Element Name | Data Type | Description or corresponding monitor element |
|-----------------------------|-----------------------|--|
| act_aborted_total | xs:nonNegativeInteger | act_aborted_total - Total aborted activities |
| act_completed_total | xs:nonNegativeInteger | act_completed_total - Total completed activities |
| act_rejected_total | xs:nonNegativeInteger | act_rejected_total - Total rejected activities |
| act_rqsts_total | xs:nonNegativeInteger | act_rqsts_total - Total activity requests |
| active_col_vector_consumers | xs:long | active_col_vector_consumers - Active columnar_vector consumers |
| active_hash_grpbys | xs:long | active_hash_grpbys - Active hash GROUP BY operations |
| active_hash_joins | xs:long | active_hash_joins - Active hash joins |
| active_olap_funcs | xs:long | active_olap_funcs - Active OLAP Functions |
| active_peas | xs:long | active_peas - Active partial early aggregations operations |
| active_peds | xs:long | active_peds - Active partial early distinct operations |
| active_sort_consumers | xs:long | active_sort_consumers - Active sort consumers |
| active_sorts | xs:long | active_sorts - Active Sorts |
| adm_bypass_act_total | xs:long | Reserved for future use. |
| adm_overflows | xs:long | Reserved for future use. |
| agent_wait_time | xs:nonNegativeInteger | agent_wait_time - Agent wait time |

Table 175. Detailed metrics returned for MON_GET_WORKLOAD_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|-------------------------------|-----------------------|--|
| agent_waits_total | xs:nonNegativeInteger | agent_waits_total - Total agent waits |
| app_act_aborted_total | xs:nonNegativeInteger | app_act_aborted_total - Total failed external coordinator activities monitor element |
| app_act_completed_total | xs:nonNegativeInteger | app_act_completed_total - Total successful external coordinator activities monitor element |
| app_act_rejected_total | xs:nonNegativeInteger | app_act_rejected_total - Total rejected external coordinator activities monitor element |
| app_rqsts_completed_total | xs:nonNegativeInteger | app_rqsts_completed_total - Total application requests completed |
| appl_section_inserts | xs:nonNegativeInteger | appl_section_inserts - Insert count for SQL workspace monitor element |
| appl_section_lookups | xs:nonNegativeInteger | appl_section_lookups - SQL workspace access count monitor element |
| audit_events_total | xs:nonNegativeInteger | audit_events_total - Total audit events |
| audit_file_write_wait_time | xs:nonNegativeInteger | audit_file_write_wait_time - Audit file write wait time |
| audit_file_writes_total | xs:nonNegativeInteger | audit_file_writes_total - Total Audit files written |
| audit_subsystem_wait_time | xs:nonNegativeInteger | audit_subsystem_wait_time - Audit subsystem wait time |
| audit_subsystem_waits_total | xs:nonNegativeInteger | audit_subsystem_waits_total - Total audit subsystem waits |
| binds_precompiles | xs:long | binds_precompiles - Binds/Precompiles Attempted monitor element |
| call_sql_stmts | xs:long | call_sql_stmts - CALL SQL statements executed monitor element |
| cat_cache_inserts | xs:nonNegativeInteger | cat_cache_inserts - Catalog cache inserts |
| cat_cache_lookups | xs:nonNegativeInteger | cat_cache_lookups - Catalog cache lookups |
| cf_wait_time | xs:long | cf_wait_time - cluster caching facility wait time |
| cf_waits | xs:long | cf_waits - Number of cluster caching facility waits |
| client_idle_wait_time | xs:nonNegativeInteger | client_idle_wait_time - Client idle wait time |
| col_synopsis_rows_inserted | xs:nonNegativeInteger | col_synopsis_rows_inserted - Column-organized synopsis table rows inserted monitor element |
| col_vector_consumer_overflows | xs:long | col_vector_consumer_overflows - Columnar vector consumer overflows monitor element. |
| comm_exit_wait_time | xs:nonNegativeInteger | comm_exit_wait_time - Communication buffer exit wait time monitor element |
| comm_exit_waits | xs:nonNegativeInteger | comm_exit_waits - Communication buffer exit number of waits monitor element |
| ddl_sql_stmts | xs:long | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| deadlocks | xs:nonNegativeInteger | deadlocks - Deadlocks detected |
| diaglog_write_wait_time | xs:nonNegativeInteger | diaglog_write_wait_time - Diag log write time |
| diaglog_writes_total | xs:nonNegativeInteger | diaglog_writes_total - Diag log total writes |
| direct_read_reqs | xs:nonNegativeInteger | direct_read_reqs - Direct read requests |
| direct_read_time | xs:nonNegativeInteger | direct_read_time - Direct read time |
| direct_reads | xs:nonNegativeInteger | direct_reads - Direct reads from database |
| direct_write_reqs | xs:nonNegativeInteger | direct_write_reqs - Direct write requests |
| direct_write_time | xs:nonNegativeInteger | direct_write_time - Direct write time |
| direct_writes | xs:nonNegativeInteger | direct_writes - Direct writes to database |
| disabled_peds | xs:long | disabled_peds - Disabled partial early distincts |
| dynamic_sql_stmts | xs:long | dynamic_sql_stmts - Dynamic SQL Statements Attempted monitor element |

Table 175. Detailed metrics returned for MON_GET_WORKLOAD_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|------------------------------|-----------------------|--|
| evmon_wait_time | xs:nonNegativeInteger | evmon_wait_time - Event monitor wait time |
| evmon_waits_total | xs:nonNegativeInteger | evmon_waits_total - Event monitor total waits |
| ext_table_read_volume | xs:nonNegativeInteger | ext_table_read_volume - Total data read by external table readers monitor element |
| ext_table_rcv_volume | xs:nonNegativeInteger | ext_table_rcv_volume - Total data received from external table readers monitor element |
| ext_table_rcv_wait_time | xs:nonNegativeInteger | ext_table_rcv_wait_time - Total agent wait time for external table readers monitor element |
| ext_table_rcvs_total | xs:nonNegativeInteger | ext_table_rcvs_total - Total row batches received from external table readers monitor element |
| ext_table_send_volume | xs:nonNegativeInteger | ext_table_send_volume - Total data sent to external table writers monitor element |
| ext_table_send_wait_time | xs:nonNegativeInteger | ext_table_send_wait_time - Total agent wait time for external table writers monitor element |
| ext_table_sends_total | xs:nonNegativeInteger | ext_table_sends_total - Total row batches sent to external table writers monitor element |
| ext_table_write_volume | xs:nonNegativeInteger | ext_table_write_volume - Total data written by external table writers monitor element |
| failed_sql_stmts | xs:long | failed_sql_stmts - Failed Statement Operations monitor element |
| fcm_message_rcv_volume | xs:nonNegativeInteger | fcm_message_rcv_volume - FCM message rcv volume |
| fcm_message_rcv_wait_time | xs:nonNegativeInteger | fcm_message_rcv_wait_time - FCM message rcv wait time |
| fcm_message_rcv_waits_total | xs:nonNegativeInteger | fcm_message_rcv_waits_total - Number of times spent waiting for FCM reply message monitor element |
| fcm_message_rcvs_total | xs:nonNegativeInteger | fcm_message_rcvs_total - FCM message rcvs total |
| fcm_message_send_volume | xs:nonNegativeInteger | fcm_message_send_volume - FCM message send volume |
| fcm_message_send_wait_time | xs:nonNegativeInteger | fcm_message_send_wait_time - FCM message send wait time |
| fcm_message_send_waits_total | xs:nonNegativeInteger | fcm_message_send_waits_total - Number of times spent blocking on an FCM message send monitor element |
| fcm_message_sends_total | xs:nonNegativeInteger | fcm_message_sends_total - FCM message sends total |
| fcm_rcv_volume | xs:nonNegativeInteger | fcm_rcv_volume - FCM rcv volume |
| fcm_rcv_wait_time | xs:nonNegativeInteger | fcm_rcv_wait_time - FCM rcv wait time |
| fcm_rcv_waits_total | xs:nonNegativeInteger | fcm_rcv_waits_total - Number of times spent waiting to receive data through FCM monitor element |
| fcm_rcvs_total | xs:nonNegativeInteger | fcm_rcvs_total - FCM rcvs total |
| fcm_send_volume | xs:nonNegativeInteger | fcm_send_volume - FCM send volume |
| fcm_send_wait_time | xs:nonNegativeInteger | fcm_send_wait_time - FCM send wait time |
| fcm_send_waits_total | xs:nonNegativeInteger | fcm_send_waits_total - Number of times spent blocking on an FCM send operation monitor element |
| fcm_sends_total | xs:nonNegativeInteger | fcm_sends_total - FCM sends total |
| fcm_tq_rcv_volume | xs:nonNegativeInteger | fcm_tq_rcv_volume - FCM tablequeue rcv volume |
| fcm_tq_rcv_wait_time | xs:nonNegativeInteger | fcm_tq_rcv_wait_time - FCM tablequeue rcv wait time |
| fcm_tq_rcv_waits_total | xs:nonNegativeInteger | fcm_tq_rcv_waits_total - Number of times spent waiting to receive the next buffer monitor element |
| fcm_tq_rcvs_total | xs:nonNegativeInteger | fcm_tq_rcvs_total - FCM tablequeue rcvs total |
| fcm_tq_send_volume | xs:nonNegativeInteger | fcm_tq_send_volume - FCM tablequeue send volume |
| fcm_tq_send_wait_time | xs:nonNegativeInteger | fcm_tq_send_wait_time - FCM tablequeue send wait time |
| fcm_tq_send_waits_total | xs:nonNegativeInteger | fcm_tq_send_waits_total - Number of times spent waiting to send the next buffer monitor element |
| fcm_tq_sends_total | xs:nonNegativeInteger | fcm_tq_sends_total - FCM tablequeue send total |

Table 175. Detailed metrics returned for MON_GET_WORKLOAD_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|---------------------------|-----------------------|---|
| fed_rows_deleted | xs:nonNegativeInteger | fed_rows_deleted - Rows deleted by a federation system monitor element |
| fed_rows_inserted | xs:nonNegativeInteger | fed_rows_inserted - Rows inserted by a federation system monitor element |
| fed_rows_read | xs:nonNegativeInteger | fed_rows_read - Rows read by a federation system monitor element |
| fed_rows_updated | xs:nonNegativeInteger | fed_rows_updated - Rows updated by a federation system monitor element |
| fed_wait_time | xs:nonNegativeInteger | fed_wait_time - Time spent by a federation server monitor element |
| fed_waits_total | xs:nonNegativeInteger | fed_waits_total - Total number of execution times for a federation server monitor element |
| hash_grpby_overflows | xs:long | hash_grpby_overflows - Hash group by overflows |
| hash_join_overflows | xs:long | hash_join_overflows - Hash Join Overflows monitor element |
| hash_join_small_overflows | xs:long | hash_join_small_overflows - Hash Join Small Overflows monitor element |
| ida_rcv_volume | xs:nonNegativeInteger | ida_rcv_volume - Total data volume received monitor element |
| ida_rcv_wait_time | xs:nonNegativeInteger | ida_rcv_wait_time - Time spent waiting to receive data monitor element |
| ida_rcvs_total | xs:nonNegativeInteger | ida_rcvs_total - Number of times data received monitor element |
| ida_send_volume | xs:nonNegativeInteger | ida_send_volume - Total data volume sent monitor element |
| ida_send_wait_time | xs:nonNegativeInteger | ida_send_wait_time - Time spent waiting to send data monitor element |
| ida_sends_total | xs:nonNegativeInteger | ida_sends_total - Number of times data sent monitor element |
| implicit_rebinds | xs:long | implicit_rebinds - Number of implicit rebinds monitor element |
| int_commits | xs:nonNegativeInteger | int_commits - Internal commits |
| int_rollback | xs:nonNegativeInteger | int_rollback - Internal rollbacks |
| int_rows_deleted | xs:long | int_rows_deleted - Internal Rows Deleted monitor element |
| int_rows_inserted | xs:long | int_rows_inserted - Internal Rows Inserted monitor element |
| int_rows_updated | xs:long | int_rows_updated - Internal Rows Updated monitor element |
| ipc_rcv_volume | xs:nonNegativeInteger | ipc_rcv_volume - Interprocess communication rcv volume |
| ipc_rcv_wait_time | xs:nonNegativeInteger | ipc_rcv_wait_time - Interprocess communication rcv wait time |
| ipc_rcvs_total | xs:nonNegativeInteger | ipc_rcvs_total - Interprocess communication rcvs total |
| ipc_send_volume | xs:nonNegativeInteger | ipc_send_volume - Interprocess communication send volume |
| ipc_send_wait_time | xs:nonNegativeInteger | ipc_send_wait_time - Interprocess communication send wait time |
| ipc_sends_total | xs:nonNegativeInteger | ipc_sends_total - Interprocess communication send total |
| lob_prefetch_reqs | xs:nonNegativeInteger | lob_prefetch_reqs - LOB prefetcher request count monitor element |
| lob_prefetch_wait_time | xs:nonNegativeInteger | lob_prefetch_wait_time - LOB prefetch wait time monitor element |
| lock_escals | xs:nonNegativeInteger | lock_escals - Number of lock escalations |
| lock_escals_global | xs:long | lock_escals_global - Number of global lock escalations |
| lock_escals_locklist | xs:long | lock_escals_locklist - Number of locklist lock escalations |
| lock_escals_maxlocks | xs:long | lock_escals_maxlocks - Number of maxlocks lock escalations |
| lock_timeouts | xs:nonNegativeInteger | lock_timeouts - Number of lock timeouts |
| lock_timeouts_global | xs:long | lock_timeouts_global - Lock timeouts global |
| lock_wait_time | xs:nonNegativeInteger | lock_wait_time - Time waited on locks |

Table 175. Detailed metrics returned for MON_GET_WORKLOAD_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|--|-----------------------|--|
| lock_wait_time_global | xs:long | lock_wait_time_global - Lock wait time global |
| lock_waits | xs:nonNegativeInteger | lock_waits - Lock waits |
| lock_waits_global | xs:long | lock_waits_global - Lock waits global |
| log_buffer_wait_time | xs:nonNegativeInteger | log_buffer_wait_time - Log buffer wait time |
| log_disk_wait_time | xs:nonNegativeInteger | log_disk_wait_time - Log disk wait time |
| log_disk_waits_total | xs:nonNegativeInteger | log_disk_waits_total - Log disk waits total |
| member | xs:nonNegativeInteger | member - Database member |
| merge_sql_stmts | xs:long | merge_sql_stmts - Merge SQL statements executed monitor element |
| num_log_buffer_full | xs:nonNegativeInteger | num_log_buffer_full - Number of full log buffers |
| num_lw_thresh_exceeded | xs:nonNegativeInteger | num_lw_thresh_exceeded - Number of thresholds exceeded |
| olap_func_overflows | xs:long | olap_func_overflows - OLAP Function Overflows monitor element |
| pkg_cache_inserts | xs:nonNegativeInteger | pkg_cache_inserts - Package cache inserts |
| pkg_cache_lookups | xs:nonNegativeInteger | pkg_cache_lookups - Package cache lookups |
| pool_col_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_col_gbp_indep_pages_found_in_lbp - Buffer pool column-organized GBP independent pages found in local buffer pool |
| pool_col_gbp_invalid_pages | xs:nonNegativeInteger | pool_col_gbp_invalid_pages - Buffer pool column-organized GBP invalid data pages |
| pool_col_gbp_l_reads | xs:nonNegativeInteger | pool_col_gbp_l_reads - Buffer pool column-organized GBP logical reads |
| pool_col_gbp_p_reads | xs:nonNegativeInteger | pool_col_gbp_p_reads - Buffer pool column-organized GBP physical reads |
| pool_col_l_reads | xs:nonNegativeInteger | pool_col_l_reads - Buffer pool column-organized logical reads |
| pool_col_lbp_pages_found | xs:nonNegativeInteger | pool_col_lbp_pages_found - Buffer pool column-organized LBP pages found |
| pool_col_p_reads | xs:nonNegativeInteger | pool_col_p_reads - Buffer pool column-organized physical reads |
| pool_col_writes | xs:nonNegativeInteger | pool_col_writes - Buffer pool column-organized writes |
| pool_data_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_data_gbp_indep_pages_found_in_lbp - Group buffer pool independent data pages found in local buffer pool monitor element |
| pool_data_gbp_invalid_pages | xs:long | pool_data_gbp_invalid_pages - Group buffer pool invalid data pages |
| pool_data_gbp_l_reads | xs:long | pool_data_gbp_l_reads - Group buffer pool data logical reads |
| pool_data_gbp_p_reads | xs:long | pool_data_gbp_p_reads - Group buffer pool data physical reads |
| pool_data_l_reads | xs:nonNegativeInteger | pool_data_l_reads - Buffer pool data logical reads |
| pool_data_lbp_pages_found | xs:long | pool_data_lbp_pages_found - Local buffer pool found data pages |
| pool_data_p_reads | xs:nonNegativeInteger | pool_data_p_reads - Buffer pool data physical reads |
| pool_data_writes | xs:nonNegativeInteger | pool_data_writes - Buffer pool data writes |
| pool_failed_async_col_reqs | xs:nonNegativeInteger | pool_failed_async_col_reqs - Failed column-organized prefetch requests |
| pool_failed_async_data_reqs | xs:nonNegativeInteger | pool_failed_async_data_reqs - Failed data prefetch requests monitor element |
| pool_failed_async_index_reqs | xs:nonNegativeInteger | pool_failed_async_index_reqs - Failed index prefetch requests monitor element |
| pool_failed_async_temp_col_reqs | xs:nonNegativeInteger | pool_failed_async_temp_col_reqs - Failed column-organized temporary prefetch requests |

Table 175. Detailed metrics returned for MON_GET_WORKLOAD_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|---|-----------------------|--|
| pool_failed_async_temp_data_reqs | xs:nonNegativeInteger | pool_failed_async_temp_data_reqs - Failed data prefetch requests for temporary table spaces monitor element |
| pool_failed_async_temp_index_reqs | xs:nonNegativeInteger | pool_failed_async_temp_index_reqs - Failed index prefetch requests for temporary table spaces monitor element |
| pool_failed_async_temp_xda_reqs | xs:nonNegativeInteger | pool_failed_async_temp_xda_reqs - Failed XDA prefetch requests for temporary table spaces monitor element |
| pool_failed_async_other_reqs | xs:nonNegativeInteger | pool_failed_async_other_reqs - Failed non-prefetch requests monitor element |
| pool_failed_async_xda_reqs | xs:nonNegativeInteger | pool_failed_async_xda_reqs - Failed XDA prefetch requests monitor element |
| pool_index_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_index_gbp_indep_pages_found_in_lbp - Group buffer pool independent index pages found in local buffer pool monitor element |
| pool_index_gbp_invalid_pages | xs:long | pool_index_gbp_invalid_pages - Group buffer pool invalid index pages |
| pool_index_gbp_l_reads | xs:long | pool_index_gbp_l_reads - Group buffer pool index logical reads |
| pool_index_gbp_p_reads | xs:long | pool_index_gbp_p_reads - Group buffer pool index physical reads |
| pool_index_l_reads | xs:nonNegativeInteger | pool_index_l_reads - Buffer pool index logical reads |
| pool_index_lbp_pages_found | xs:long | pool_index_lbp_pages_found - Local buffer pool index pages found |
| pool_index_p_reads | xs:nonNegativeInteger | pool_index_p_reads - Buffer pool index physical reads |
| pool_index_writes | xs:nonNegativeInteger | pool_index_writes - Buffer pool index writes |
| pool_queued_async_col_reqs | xs:nonNegativeInteger | pool_queued_async_col_reqs - Column-organized prefetch requests |
| pool_queued_async_data_pages | xs:nonNegativeInteger | pool_queued_async_data_pages - Data pages prefetch requests monitor element |
| pool_queued_async_data_reqs | xs:nonNegativeInteger | pool_queued_async_data_reqs - Data prefetch requests monitor element |
| pool_queued_async_index_reqs | xs:nonNegativeInteger | pool_queued_async_index_reqs - Index prefetch requests monitor element |
| pool_queued_async_col_pages | xs:nonNegativeInteger | pool_queued_async_col_pages - Column-organized page prefetch requests |
| pool_queued_async_index_pages | xs:nonNegativeInteger | pool_queued_async_index_pages - Index pages prefetch requests monitor element |
| pool_queued_async_temp_col_pages | xs:nonNegativeInteger | pool_queued_async_temp_col_pages - Column-organized page temporary prefetch requests |
| pool_queued_async_temp_col_reqs | xs:nonNegativeInteger | pool_queued_async_temp_col_reqs - Column-organized temporary prefetch requests |
| pool_queued_async_temp_data_pages | xs:nonNegativeInteger | pool_queued_async_temp_data_pages - Data pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_data_reqs | xs:nonNegativeInteger | pool_queued_async_temp_data_reqs - Data prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_index_pages | xs:nonNegativeInteger | pool_queued_async_temp_index_pages - Index pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_index_reqs | xs:nonNegativeInteger | pool_queued_async_temp_index_reqs - Index prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_xda_pages | xs:nonNegativeInteger | pool_queued_async_temp_xda_pages - XDA data pages prefetch requests for temporary table spaces monitor element |
| pool_queued_async_temp_xda_reqs | xs:nonNegativeInteger | pool_queued_async_temp_xda_reqs - XDA data prefetch requests for temporary table spaces monitor element |
| pool_queued_async_other_reqs | xs:nonNegativeInteger | pool_queued_async_other_reqs - Non-prefetch requests monitor element |

Table 175. Detailed metrics returned for MON_GET_WORKLOAD_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|---------------------------------------|-----------------------|--|
| pool_queued_async_xda_pages | xs:nonNegativeInteger | pool_queued_async_xda_pages - XDA pages prefetch requests monitor element |
| pool_queued_async_xda_reqs | xs:nonNegativeInteger | pool_queued_async_xda_reqs - XDA prefetch requests monitor element |
| pool_read_time | xs:nonNegativeInteger | pool_read_time - Total buffer pool physical read time |
| pool_temp_col_l_reads | xs:nonNegativeInteger | pool_temp_col_l_reads - Buffer pool column-organized temporary logical reads |
| pool_temp_col_p_reads | xs:nonNegativeInteger | pool_temp_col_p_reads - Buffer pool column-organized temporary physical reads |
| pool_temp_data_l_reads | xs:nonNegativeInteger | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| pool_temp_data_p_reads | xs:nonNegativeInteger | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| pool_temp_index_l_reads | xs:nonNegativeInteger | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| pool_temp_index_p_reads | xs:nonNegativeInteger | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| pool_temp_xda_l_reads | xs:nonNegativeInteger | pool_temp_xda_l_reads - Buffer pool temporary XDA data logical reads |
| pool_temp_xda_p_reads | xs:nonNegativeInteger | pool_temp_xda_p_reads - Buffer pool temporary XDA data physical reads |
| pool_write_time | xs:nonNegativeInteger | pool_write_time - Total buffer pool physical write time |
| pool_xda_gbp_indep_pages_found_in_lbp | xs:nonNegativeInteger | pool_xda_gbp_indep_pages_found_in_lbp - Group buffer pool XDA independent pages found in local buffer pool monitor element |
| pool_xda_gbp_invalid_pages | xs:nonNegativeInteger | pool_xda_gbp_invalid_pages - Group buffer pool invalid XDA data pages |
| pool_xda_gbp_l_reads | xs:nonNegativeInteger | pool_xda_gbp_l_reads - Group buffer pool XDA data logical read requests |
| pool_xda_gbp_p_reads | xs:nonNegativeInteger | pool_xda_gbp_p_reads - Group buffer pool XDA data physical read requests |
| pool_xda_l_reads | xs:nonNegativeInteger | pool_xda_l_reads - Buffer pool XDA data logical reads |
| pool_xda_lbp_pages_found | xs:nonNegativeInteger | pool_xda_lbp_pages_found - Local buffer pool XDA data pages found |
| pool_xda_p_reads | xs:nonNegativeInteger | pool_xda_p_reads - Buffer pool XDA data physical reads |
| pool_xda_writes | xs:nonNegativeInteger | pool_xda_writes - Buffer pool XDA data writes |
| post_shrthreshold_hash_joins | xs:long | post_shrthreshold_hash_joins - Post threshold hash joins monitor element |
| post_shrthreshold_sorts | xs:nonNegativeInteger | post_shrthreshold_sorts - Post shared threshold sorts |
| post_threshold_col_vector_consumers | xs:long | post_threshold_col_vector_consumers - Post threshold columnar_vector consumers |
| post_threshold_hash_grpbys | xs:long | post_threshold_hash_grpbys - Hash group by threshold |
| post_threshold_hash_joins | xs:long | post_threshold_hash_joins - Hash Join Threshold monitor element |
| post_threshold_olap_funcs | xs:long | post_threshold_olap_funcs - OLAP function threshold monitor element |
| post_threshold_peas | xs:long | post_threshold_peas - Partial early aggregation threshold |
| post_threshold_peds | xs:long | post_threshold_peds - Partial early distincts threshold |
| post_threshold_sorts | xs:nonNegativeInteger | post_threshold_sorts - Post threshold sorts |
| prefetch_wait_time | xs:nonNegativeInteger | prefetch_wait_time - Time waited for prefetch |
| prefetch_waits | xs:nonNegativeInteger | prefetch_waits - Prefetcher wait count monitor element |

Table 175. Detailed metrics returned for MON_GET_WORKLOAD_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|--------------------------------|-----------------------|--|
| reclaim_wait_time | xs:long | reclaim_wait_time - Reclaim wait time |
| rows_deleted | xs:long | rows_deleted - Rows deleted monitor element |
| rows_inserted | xs:long | rows_inserted - Rows inserted monitor element |
| rows_modified | xs:nonNegativeInteger | rows_modified - Rows modified |
| rows_read | xs:nonNegativeInteger | rows_read - Rows read |
| rows_returned | xs:nonNegativeInteger | rows_returned - Rows returned |
| rows_updated | xs:long | rows_updated - Rows updated monitor element |
| rqsts_completed_total | xs:nonNegativeInteger | rqsts_completed_total - Total requests completed |
| select_sql_stmts | xs:long | select_sql_stmts - Select SQL Statements Executed monitor element |
| sort_heap_allocated | xs:long | sort_heap_allocated - Total Sort Heap Allocated |
| sort_overflows | xs:nonNegativeInteger | sort_overflows - Sort overflows |
| sort_shrheap_allocated | xs:long | sort_shrheap_allocated - Sort Share Heap Currently Allocated |
| spacemappage_reclaim_wait_time | xs:long | spacemappage_reclaim_wait_time - Space map page reclaim wait time |
| static_sql_stmts | xs:long | static_sql_stmts - Static SQL Statements Attempted monitor element |
| tcpip_rcv_volume | xs:nonNegativeInteger | tcpip_rcv_volume - TCP/IP received volume |
| tcpip_rcv_wait_time | xs:nonNegativeInteger | tcpip_rcv_wait_time - TCP/IP rcv wait time |
| tcpip_rcvs_total | xs:nonNegativeInteger | tcpip_rcvs_total - TCP/IP rcvs total |
| tcpip_send_volume | xs:nonNegativeInteger | tcpip_send_volume - TCP/IP send volume |
| tcpip_send_wait_time | xs:nonNegativeInteger | tcpip_send_wait_time - TCP/IP send wait time |
| tcpip_sends_total | xs:nonNegativeInteger | tcpip_sends_total - TCP/IP sends total |
| thresh_violations | xs:nonNegativeInteger | thresh_violations - Number of threshold violations |
| total_act_time | xs:nonNegativeInteger | total_act_time - Total activity time |
| total_act_wait_time | xs:nonNegativeInteger | total_act_wait_time - Total activity wait time |
| total_app_commits | xs:nonNegativeInteger | total_app_commits - Total application commits |
| total_app_rollback | xs:nonNegativeInteger | total_app_rollback - Total application rollbacks |
| total_app_rqst_time | xs:nonNegativeInteger | total_app_rqst_time - Total application request time |
| total_app_section_executions | xs:nonNegativeInteger | total_app_section_executions - Total section executions |
| total_backup_proc_time | xs:long | total_backup_proc_time - Total non-wait time for online backups |
| total_backup_time | xs:long | total_backup_time - Total elapsed time for online backups |
| total_backups | xs:long | total_backups - Total backups |
| total_col_executions | xs:nonNegativeInteger | total_col_executions - Total column-organized executions |
| total_col_proc_time | xs:nonNegativeInteger | total_col_proc_time - Total column-organized processing time |
| total_col_synopsis_executions | xs:nonNegativeInteger | total_col_synopsis_executions - Total column-organized synopsis table executions monitor element |
| total_col_synopsis_proc_time | xs:nonNegativeInteger | total_col_synopsis_proc_time - Total column-organized synopsis table processing time monitor element |
| total_col_synopsis_time | xs:nonNegativeInteger | total_col_synopsis_time - Total column-organized synopsis table time monitor element |
| total_col_time | xs:nonNegativeInteger | total_col_time - Total column-organized time |
| total_col_vector_consumers | xs:long | total_col_vector_consumers - Total columnar_vector consumers |
| total_commit_proc_time | xs:nonNegativeInteger | total_commit_proc_time - Total commits processing time |
| total_commit_time | xs:nonNegativeInteger | total_commit_time - Total commit time |
| total_compilations | xs:nonNegativeInteger | total_compilations - Total compilations |

Table 175. Detailed metrics returned for MON_GET_WORKLOAD_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|--|-----------------------|--|
| total_compile_proc_time | xs:nonNegativeInteger | total_compile_proc_time - Total compile processing time |
| total_compile_time | xs:nonNegativeInteger | total_compile_time - Total compile time |
| total_connect_authentication_proc_time | xs:nonNegativeInteger | total_connect_authentication_proc_time - Total connection authentication processing time |
| total_connect_authentication_time | xs:nonNegativeInteger | total_connect_authentication_time - Total connection or switch user authentication request time |
| total_connect_authentications | xs:nonNegativeInteger | total_connect_authentications - Connections or switch user authentications performed |
| total_connect_request_proc_time | xs:nonNegativeInteger | total_connect_request_proc_time - Total connection or switch user request processing time |
| total_connect_request_time | xs:nonNegativeInteger | total_connect_request_time - Total connection or switch user request time |
| total_connect_requests | xs:nonNegativeInteger | total_connect_requests - Connection or switch user requests |
| total_cpu_time | xs:nonNegativeInteger | total_cpu_time - Total CPU time |
| total_disp_run_queue_time | xs:long | total_disp_run_queue_time - Total dispatcher run queue time |
| total_extended_latch_wait_time | xs:nonNegativeInteger | total_extended_latch_wait_time - Total extended latch wait time |
| total_extended_latch_waits | xs:nonNegativeInteger | total_extended_latch_waits - Total extended latch waits |
| total_hash_grpbys | xs:long | total_hash_grpbys - Total hash group by operations |
| total_hash_joins | xs:long | total_hash_joins - Total Hash Joins monitor element |
| total_hash_loops | xs:long | total_hash_loops - Total Hash Loops monitor element |
| total_implicit_compilations | xs:nonNegativeInteger | total_implicit_compilations - Total implicit complications |
| total_implicit_compile_proc_time | xs:nonNegativeInteger | total_implicit_compile_proc_time - Total implicit compile processing time |
| total_implicit_compile_time | xs:nonNegativeInteger | total_implicit_compile_time - Total implicit compile time |
| total_index_build_proc_time | xs:long | total_index_build_proc_time - Total non-wait time spent building indexes due to index creation, recreation or rebuilding |
| total_index_build_time | xs:long | total_index_build_time - Total time spent building indexes due to index creation, recreation or rebuilding |
| total_indexes_built | xs:long | total_indexes_built - Total number of indexes built |
| total_load_proc_time | xs:nonNegativeInteger | total_load_proc_time - Total load processing time |
| total_load_time | xs:nonNegativeInteger | total_load_time - Total load time |
| total_loads | xs:nonNegativeInteger | total_loads - Total loads |
| total_olap_funcs | xs:long | total_olap_funcs - Total OLAP Functions monitor element |
| total_peas | xs:long | total_peas - Total partial early aggregations |
| total_peds | xs:long | total_peds - Total partial early distincts |
| total_reorg_proc_time | xs:nonNegativeInteger | total_reorg_proc_time - Total reorganization processing time |
| total_reorg_time | xs:nonNegativeInteger | total_reorg_time - Total reorganization time |
| total_reorgs | xs:nonNegativeInteger | total_reorgs - Total reorganizations |
| total_rollback_proc_time | xs:nonNegativeInteger | total_rollback_proc_time - Total rollback processing time |
| total_rollback_time | xs:nonNegativeInteger | total_rollback_time - Total rollback time |
| total_routine_invocations | xs:nonNegativeInteger | total_routine_invocations - Total routine invocations |
| total_routine_time | xs:nonNegativeInteger | total_routine_time - Total routine time |
| total_routine_user_code_proc_time | xs:nonNegativeInteger | total_routine_user_code_proc_time - Total routine user code processing time |
| total_routine_user_code_time | xs:nonNegativeInteger | total_routine_user_code_time - Total routine user code time |
| total_rqst_time | xs:nonNegativeInteger | total_rqst_time - Total request time |
| total_runstats | xs:nonNegativeInteger | total_runstats - Total runtime statistics |

Table 175. Detailed metrics returned for MON_GET_WORKLOAD_DETAILS (continued)

| Element Name | Data Type | Description or corresponding monitor element |
|-----------------------------------|-----------------------|--|
| total_runstats_proc_time | xs:nonNegativeInteger | total_runstats_proc_time - Total runtime statistics processing time |
| total_runstats_time | xs:nonNegativeInteger | total_runstats_time - Total runtime statistics |
| total_section_proc_time | xs:nonNegativeInteger | total_section_proc_time - Total section processing time |
| total_section_sort_proc_time | xs:nonNegativeInteger | total_section_sort_proc_time - Total section sort processing time |
| total_section_sort_time | xs:nonNegativeInteger | total_section_sort_time - Total section sort time |
| total_section_sorts | xs:nonNegativeInteger | total_section_sorts - Total section sorts |
| total_section_time | xs:nonNegativeInteger | total_section_time - Total section time |
| total_sorts | xs:nonNegativeInteger | total_sorts - Total Sorts |
| total_stats_fabrication_proc_time | xs:nonNegativeInteger | total_stats_fabrication_proc_time - Total statistics fabrication processing time |
| total_stats_fabrication_time | xs:nonNegativeInteger | total_stats_fabrication_time - Total statistics fabrication time |
| total_stats_fabrications | xs:nonNegativeInteger | total_stats_fabrications - Total statistics fabrications |
| total_sync_runstats | xs:nonNegativeInteger | total_sync_runstats - Total synchronous RUNSTATS activities |
| total_sync_runstats_proc_time | xs:nonNegativeInteger | total_sync_runstats_proc_time - Total synchronous RUNSTATS processing time |
| total_sync_runstats_time | xs:nonNegativeInteger | total_sync_runstats_time - Total synchronous RUNSTATS time |
| total_wait_time | xs:nonNegativeInteger | total_wait_time - Total wait time |
| tq_sort_heap_rejections | xs:long | tq_sort_heap_rejections - Table queue sort heap rejections |
| tq_sort_heap_requests | xs:long | tq_sort_heap_requests - Table queue sort heap requests |
| tq_tot_send_spills | xs:nonNegativeInteger | tq_tot_send_spills - Total number of table queue buffers overflowed |
| uid_sql_stmts | xs:long | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| wlm_queue_assignments_total | xs:nonNegativeInteger | wlm_queue_assignments_total - Workload manager total queue assignments |
| wlm_queue_time_total | xs:nonNegativeInteger | wlm_queue_time_total - Workload manager total queue time |
| workload_id | xs:nonNegativeInteger | workload_id - Workload ID |
| workload_name | xs:string (128) | workload_name - Workload name |
| xquery_stmts | xs:long | xquery_stmts - XQuery Statements Attempted monitor element |

Usage notes

The metrics returned by the MON_GET_WORKLOAD_DETAILS function represent the accumulation of all metrics for requests that were submitted by connections mapped to the identified workload object. This function is similar to the MON_GET_WORKLOAD table function:

- The MON_GET_WORKLOAD table function returns the most commonly used metrics in a column-based format and is the most performance efficient method of retrieving metrics.
- The MON_GET_WORKLOAD_DETAILS table function returns the entire set of available metrics in an XML document format, which provides maximum flexibility for formatting output. The XML based output can be parsed directly by an XML parser, or it can be converted to relational format by the XMLTABLE function (see the example).

Metrics are rolled up to a workload on unit of work boundaries, and periodically during the execution of requests. Therefore, the values reported by this table function reflect the current state of the system at the time of the most recent rollup.

Metrics are strictly increasing in value. To determine the value of a given metric for an interval of time, use the MON_GET_WORKLOAD_DETAILS table function to query the metric at the start and end of the interval, and compute the difference.

Request metrics are controlled through the COLLECT REQUEST METRICS clause on service superclasses and the **mon_req_metrics** database configuration parameter at the database level. Metrics are only collected for a request if the request is processed by an agent in a service subclass whose parent service superclass has request metrics enabled, or if request metrics collection is enabled for the entire database. By default request metrics are enabled at the database level. If request metrics have been disabled at the database level, and for a service superclass, the metrics reported for each workload mapped to that service superclass stop increasing (or remain at 0 if request metrics were disabled at database activation time).

Note that the metrics from background EDUs such as prefetchers and page cleaners are not rolled up to workloads.

The MON_GET_WORKLOAD_DETAILS table function returns one row of data per workload and per member. No aggregation across workloads (on a member), or across members (for a service class or more), is performed. However, aggregation can be achieved through SQL queries as shown in the example.

The schema for the XML document that is returned in the DETAILS column is available in the file sqllib/misc/DB2MonRoutines.xsd. Further details can be found in the file sqllib/misc/DB2MonCommon.xsd.

Example

Display lock information for each workload, aggregated across members, ordered by highest lock wait time.

```
SELECT varchar(wlmetrics.workload_name,30) as workload_name,
       sum(detmetrics.lock_wait_time) as total_lock_wait_time,
       sum(detmetrics.lock_waits) as total_lock_waits,
       sum(detmetrics.lock_timeouts) as total_lock_timeouts,
       sum(detmetrics.lock_escals) as total_lock_escals
FROM TABLE(MON_GET_WORKLOAD_DETAILS('','-2')) AS WLMETRICS,
XMLTABLE (XMLNAMESPACES( DEFAULT 'http://www.ibm.com/xmlns/prod/db2/mon'),
          '$detmetric/db2_workload' PASSING
          XMLPARSE(DOCUMENT WLMETRICS.DETAILS)
          as "detmetric"
COLUMNS "LOCK_WAIT_TIME" INTEGER PATH 'system_metrics/lock_wait_time',
         "LOCK_WAITS" INTEGER PATH 'system_metrics/lock_waits',
         "LOCK_TIMEOUTS" INTEGER PATH 'system_metrics/lock_timeouts',
         "LOCK_ESCALS" INTEGER PATH 'system_metrics/lock_escals'
) AS DETMETRICS
GROUP BY workload_name
ORDER BY total_lock_wait_time desc;
```

The following is an example of output from this query.

| WORKLOAD_NAME | TOTAL_LOCK_WAIT_TIME | TOTAL_LOCK_WAITS | ... |
|------------------------|----------------------|------------------|-----|
| SYSDEFAULTADMWORKLOAD | 0 | 0 | ... |
| SYSDEFAULTUSERWORKLOAD | 0 | 0 | ... |

2 record(s) selected.

Output for query (continued).

```

... TOTAL_LOCK_TIMEOUTS TOTAL_LOCK_ESCALS
... -----
... 0 0
... 0 0

```

MON_GET_WORKLOAD_STATS table function - Return workload statistics

The MON_GET_WORKLOAD_STATS function returns one row of workload statistics for every combination of workload name and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Syntax

```

▶▶ MON_GET_WORKLOAD_STATS (—workload_name—, —member—) ▶▶

```

The schema is SYSPROC.

Routine parameters

workload_name

An input argument of type VARCHAR(128) that specifies a workload for which the statistics are to be returned. If the argument is NULL or an empty string, statistics are returned for all workloads.

member

An input argument of type INTEGER that specifies the number of a member in the same instance as the currently connected database. Specify -1 for the current member, or -2 for all active database members. If a null value is specified, -1 is set implicitly.

Information returned

Table 176. Information returned by MON_GET_WORKLOAD_STATS

| Column name | Data type | Description |
|------------------------|--------------|--|
| WORKLOAD_NAME | VARCHAR(128) | workload_name - Workload name monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |
| CONCURRENT_WLO_TOP | INTEGER | concurrent_wlo_top - Concurrent workload occurrences top monitor element |
| CONCURRENT_WLO_ACT_TOP | INTEGER | concurrent_wlo_act_top - Concurrent WLO activity top monitor element |

Table 176. Information returned by MON_GET_WORKLOAD_STATS (continued)

| Column name | Data type | Description |
|-----------------------------|-----------|---|
| COORD_ACT_COMPLETED_TOTAL | BIGINT | coord_act_completed_total - Coordinator activities completed total monitor element |
| COORD_ACT_ABORTED_TOTAL | BIGINT | coord_act_aborted_total - Coordinator activities aborted total monitor element |
| COORD_ACT_REJECTED_TOTAL | BIGINT | coord_act_rejected_total - Coordinator activities rejected total monitor element |
| WLO_COMPLETED_TOTAL | BIGINT | wlo_completed_total - Workload occurrences completed total monitor element |
| COORD_ACT_LIFETIME_TOP | BIGINT | coord_act_lifetime_top - Coordinator activity lifetime top monitor element |
| COORD_ACT_LIFETIME_AVG | DOUBLE | coord_act_lifetime_avg - Coordinator activity lifetime average monitor element |
| COORD_ACT_LIFETIME_STDDEV | DOUBLE | Standard deviation of lifetime for completed or aborted coordinator activities at nesting level 0 that are associated with this workload. Units are milliseconds. If the COLLECT AGGREGATE ACTIVITY DATA parameter of the workload is set to NONE, the value of the column is null. This standard deviation is computed from the coordinator activity lifetime histogram and may be inaccurate if the histogram is not correctly sized to fit the data. If any values fall into the last histogram bin, the value -1 is returned. |
| COORD_ACT_EXEC_TIME_AVG | DOUBLE | coord_act_exec_time_avg - Coordinator activities execution time average monitor element |
| COORD_ACT_EXEC_TIME_STDDEV | DOUBLE | Standard deviation of the execution times for completed or aborted coordinator activities at nesting level 0 that are associated with this workload. Units are milliseconds. This standard deviation is computed from the coordinator activity executetime histogram and may be inaccurate if the histogram is not correctly sized to fit the data. If any values fall into the last histogram bin, the value -1 is returned. If the COLLECT AGGREGATE ACTIVITY DATA parameter of the workload is set to NONE, the value of the column is null. |
| COORD_ACT_QUEUE_TIME_AVG | DOUBLE | coord_act_queue_time_avg - Coordinator activity queue time average monitor element |
| COORD_ACT_QUEUE_TIME_STDDEV | DOUBLE | Standard deviation of the queue time for completed or aborted coordinator activities at nesting level 0 that are associated with this workload. Units are milliseconds. If the COLLECT AGGREGATE ACTIVITY DATA parameter of the workload is set to NONE, the value of the column is null. This standard deviation is computed from the coordinator activity queue time histogram and may be inaccurate if the histogram is not correctly sized to fit the data. If any values fall into the last histogram bin, the value -1 is returned. |
| UOW_TOTAL_TIME_TOP | BIGINT | uow_total_time_top - UOW total time top monitor element |

Table 176. Information returned by MON_GET_WORKLOAD_STATS (continued)

| Column name | Data type | Description |
|---------------------------------|-----------|--|
| UOW_THROUGHPUT | DOUBLE | uow_throughput - Unit of work throughput monitor element |
| UOW_LIFETIME_AVG | DOUBLE | uow_lifetime_avg - Unit of work lifetime average monitor element |
| UOW_COMPLETED_TOTAL | BIGINT | uow_completed_total - Total completed units of work monitor element |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time monitor element |
| TOTAL_DISP_RUN_QUEUE_TIME | BIGINT | total_disp_run_queue_time - Total dispatcher run queue time monitor element |
| ACT_THROUGHPUT | DOUBLE | act_throughput - Activity throughput monitor element |
| CPU_UTILIZATION | DOUBLE | cpu_utilization - CPU utilization monitor element |
| APP_ACT_COMPLETED_TOTAL | BIGINT | app_act_completed_total - Total successful external coordinator activities monitor element |
| APP_ACT_ABORTED_TOTAL | BIGINT | app_act_aborted_total - Total failed external coordinator activities monitor element |
| APP_ACT_REJECTED_TOTAL | BIGINT | app_act_rejected_total - Total rejected external coordinator activities monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| ACTIVE_HASH_GRPBY_TOP | BIGINT | active_hash_grpbys_top - Active hash Group By operations high watermark |
| ACTIVE_HASH_JOINS_TOP | BIGINT | active_hash_joins_top - Active hash joins operations high watermark |
| ACTIVE_OLAP_FUNCS_TOP | BIGINT | active_olap_funcs_top - Active OLAP functions operations high watermark |
| ACTIVE_PEAS_TOP | BIGINT | active_peas_top - Active partial early aggregations operations high watermark |
| ACTIVE_PEDS_TOP | BIGINT | active_peds_top - Active partial early distinct operations high watermark |
| ACTIVE_SORT_CONSUMERS_TOP | BIGINT | active_sort_consumers_top - Active sort consumers high watermark |
| ACTIVE_SORTS_TOP | BIGINT | active_sorts_top - Active Sorts high watermark |
| ACTIVE_COL_VECTOR_CONSUMERS_TOP | BIGINT | active_col_vector_consumers_top - Active columnar_vector consumers high watermark |
| SORT_CONSUMER_HEAP_TOP | BIGINT | sort_consumer_heap_top - Individual private sort heap consumer high watermark |
| SORT_CONSUMER_SHRHEAP_TOP | BIGINT | sort_consumer_shrheap_top - Individual shared sort heap consumer high watermark |
| SORT_HEAP_TOP | BIGINT | sort_heap_top - Sort private heap high watermark |
| SORT_SHRHEAP_TOP | BIGINT | sort_shrheap_top - Sort share heap high watermark |

Usage notes

The function does not aggregate data across workloads, members, or service classes. However, you can use SQL queries to aggregate data.

Example

The following query displays statistics for workloads:

```
SELECT SUBSTR(WORKLOAD_NAME,1,18) AS WL_DEF_NAME,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       COORD_ACT_LIFETIME_TOP,
       COORD_ACT_LIFETIME_AVG,
       COORD_ACT_LIFETIME_STDDEV
FROM TABLE(MON_GET_WORKLOAD_STATS(CAST(NULL AS VARCHAR(128)), -2)) AS WLSTATS
ORDER BY WL_DEF_NAME, MEMB
```

Sample output from the query is as follows:

| WL_DEF_NAME | MEMB | COORD_ACT_LIFETIME_TOP | ... |
|--------------------|-------------------------|---------------------------|-----|
| SYSDEFAULTADMWORKL | 0 | -1 | ... |
| SYSDEFAULTUSERWORK | 0 | -1 | ... |
| WL1 | 0 | 2 | ... |
| ... | COORD_ACT_LIFETIME_AVG | COORD_ACT_LIFETIME_STDDEV | |
| ... | -1.000000000000000E+000 | -1.000000000000000E+000 | |
| ... | -1.000000000000000E+000 | -1.000000000000000E+000 | |
| ... | +2.560000000000000E+000 | +6.00000000000001E-002 | |

MON_GET_WORK_ACTION_SET_STATS - Return work action set statistics

The MON_GET_WORK_ACTION_SET_STATS function returns the statistics for a work action set.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_GET_WORK_ACTION_SET_STATS—(—*work_action_set_name*—,—*member*—)—►◄

The schema is SYSPROC.

Routine parameters

work_action_set_name

An input argument of type VARCHAR(128) that specifies the work action set to return statistics for. If the argument is null or an empty string, statistics are returned for all work action sets.

member

An input argument of type INTEGER that specifies a valid member number in

the same instance as the currently connected database. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

Information returned

Table 177. Information returned for MON_GET_WORK_ACTION_SET_STATS

| Column name | Data type | Description |
|----------------------|--------------|---|
| WORK_ACTION_SET_NAME | VARCHAR(128) | work_action_set_name - Work action set name monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |
| WORK_CLASS_NAME | VARCHAR(128) | work_class_name - Work class name monitor element |
| ACT_TOTAL | BIGINT | act_total - Activities total monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

Example

Assume that there are three work classes: ReadClass, WriteClass, and LoadClass. There is a work action associated with ReadClass and a work action associated with LoadClass, but there is no work action associated with WriteClass. On member 0, the number of activities currently running or queued are as follows:

- ReadClass class: eight
- WriteClass class: four
- LoadClass class: two
- Unassigned: three

```
SELECT SUBSTR(WORK_ACTION_SET_NAME,1,18) AS WORK_ACTION_SET_NAME,
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,
       SUBSTR(WORK_CLASS_NAME,1,15) AS WORK_CLASS_NAME,
       LAST_RESET,
       SUBSTR(CHAR(ACT_TOTAL),1,14) AS ACT_TOTAL
FROM TABLE(MON_GET_WORK_ACTION_SET_STATS
             (CAST(NULL AS VARCHAR(128)), -2)) AS WASSTATS
ORDER BY WORK_ACTION_SET_NAME, WORK_CLASS_NAME, MEMB
```

Sample output is as follows. Because there is no work action associated with the WriteClass work class, the four activities to which it applies are counted in the artificial class denoted by an asterisk (*) in the output. The three activities that were not assigned to any work class are also included in the artificial class.

| WORK_ACTION_SET_NAME | MEMB | WORK_CLASS_NAME | LAST_RESET | ACT_TOTAL |
|----------------------|------|-----------------|----------------------------|-----------|
| AdminActionSet | 0 | ReadClass | 2005-11-25-18.52.49.343000 | 8 |
| AdminActionSet | 1 | ReadClass | 2005-11-25-18.52.50.478000 | 0 |
| AdminActionSet | 0 | LoadClass | 2005-11-25-18.52.49.343000 | 2 |
| AdminActionSet | 1 | LoadClass | 2005-11-25-18.52.50.478000 | 0 |
| AdminActionSet | 0 | * | 2005-11-25-18.52.49.343000 | 7 |
| AdminActionSet | 1 | * | 2005-11-25-18.52.50.478000 | 0 |

MON_INCREMENT_INTERVAL_ID procedure - increment the monitoring interval

The MON_INCREMENT_INTERVAL_ID procedure increments the monitoring interval by 1 and returns the new value in the output argument.

The current monitoring interval is indicated by the MON_INTERVAL_ID database global variable.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

►►—MON_INTERVAL_INCREMENT_ID—(*—new_id—*)—►►

The schema is SYSPROC.

Procedure parameters

new_id

An output argument of type BIGINT that returns the new monitoring interval.

Usage notes

As an autonomous procedure, MON_INCREMENT_INTERVAL_ID executes in its own transaction scope.

When the value of MON_INTERVAL_ID reaches the maximum BIGINT value, it is cycled back to 1.

Example

To increment the interval ID and return the new value in the argument. This example assumes that MY_ID is a previously created session global variable.

```
CALL SYSPROC.MON_INCREMENT_INTERVAL_ID(MY_ID)
```

MON_SAMPLE_SERVICE_CLASS_METRICS - Get sample service class metrics

The MON_SAMPLE_SERVICE_CLASS_METRICS table function reads system metrics for one or more service classes across one or more databases at two points in time: at the time the function is called and after a given amount of time has passed.

The function computes various statistics from these metrics.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority

- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

```
►►MON_SAMPLE_SERVICE_CLASS_METRICS—(—hostname—,—db_name—,——————►
►service_superclass_name—,—service_subclass_name—,——————►
►sample_time—,—member—)——————►◄
```

The schema is SYSPROC.

Table function parameters

hostname

An input argument of type VARCHAR(255) that specifies a fully qualified host name or an IP address of the node from which to generate the report. If the argument is NULL or an empty string, metrics are taken from all active databases in the instance.

db_name

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database when calling this function. The database must have a directory entry type of either "Indirect" or "Home", as returned by a LIST DATABASE DIRECTORY command. If the argument is NULL or an empty string, metrics are taken from all databases in the instance.

service_superclass_name

An input argument of type VARCHAR(128) that specifies a valid service superclass name in the currently connected database when calling this function. If the argument is NULL or an empty string, performance metrics are retrieved for all the superclasses in the instance whose database name satisfies the constraint imposed by the db_name parameter.

service_subclass_name

An input argument of type VARCHAR(128) that specifies a valid service subclass name in the currently connected database when calling this function. If the argument is NULL or an empty string, performance metrics are retrieved for all the subclasses in the instance whose database name and service superclass name satisfy the constraints imposed by the db_name and service_superclass_name parameters, respectively.

sample_time

The amount of time the function collects data before computing a result on that data. This value is measured in seconds and must be greater than or equal to 1.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all database members. If the NULL value is specified, -1 is set implicitly.

Information returned

Table 178. Information returned for MON_SAMPLE_SERVICE_CLASS_METRICS

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|--------------|---|
| HOSTNAME | VARCHAR(255) | hostname - Host name monitor element |
| DB_NAME | VARCHAR(128) | db_name - Database name monitor element |
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name |
| SERVICE_SUBCLASS_NAME | VARCHAR(128) | service_subclass_name - Service subclass name |
| SERVICE_CLASS_ID | INTEGER | service_class_id - Service class ID |
| MEMBER | SMALLINT | member - Database member |
| UOW_THROUGHPUT | DOUBLE | uow_throughput - Unit of work throughput monitor element |
| UOW_LIFETIME_AVG | DOUBLE | uow_lifetime_avg - Unit of work lifetime average monitor element |
| UOW_COMPLETED_TOTAL | BIGINT | uow_completed_total - Total completed units of work monitor element |
| ACT_THROUGHPUT | DOUBLE | act_throughput - Activity throughput monitor element |
| ACT_COMPLETED_TOTAL | BIGINT | act_completed_total - Total completed activities |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time monitor element |
| TOTAL_DISP_RUN_QUEUE_TIME | BIGINT | total_disp_run_queue_time - Total dispatcher run queue time monitor element |
| CPU_SHARES | INTEGER | cpu_shares - WLM dispatcher CPU shares monitor element |
| CPU_SHARE_TYPE | VARCHAR(4) | cpu_share_type - WLM dispatcher CPU share type monitor element |
| CPU_LIMIT | SMALLINT | cpu_limit - WLM dispatcher CPU limit monitor element |
| ESTIMATED_CPU_ENTITLEMENT | DOUBLE | estimated_cpu_entitlement - Estimated CPU entitlement monitor element |
| CPU_UTILIZATION | DOUBLE | cpu_utilization - CPU utilization monitor element |
| CPU_VELOCITY | DOUBLE | cpu_velocity - CPU velocity monitor element |
| ADAPTIVE_SHARES_ENABLED | BOOLEAN | Reserved for future use. |

Usage notes

The MON_SAMPLE_SERVICE_CLASS_METRICS table function returns one row of data for each service class in each active database, per member.

The MON_SAMPLE_SERVICE_CLASS_METRICS table function also returns an additional row of output for each active database, per member. These additional rows show the CPU entitlement and usage for each database as a whole. In the database rows, all fields are populated, except for the service class identifier columns, whose values are NULL:

- SERVICE_SUPERCLASS_NAME
- SERVICE_SUBCLASS_NAME
- SERVICE_CLASS_ID

The table function performs no aggregation across service classes (on a member), or across members (for a service class or more). However, aggregation can be achieved through SQL queries. The input parameters have the effect of being “ANDed” together. Therefore, if you specify conflicting input parameters (for example, a superclass named SUPA, and subclass named SUBB which is not a subclass of SUPA), then no rows are returned.

Example 1

Show the activity throughput and CPU velocity for a 30 second period for each service subclass across all partitions.

```
SELECT varchar(service_superclass_name,30) AS service_superclass,  
       varchar(service_subclass_name,30) AS service_subclass,  
       decimal(sum(act_throughput),10,2) AS act_throughput,  
       decimal(sum(total_cpu_time) /  
              (sum(total_cpu_time) +  
               sum(total_disp_run_queue_time)),3,2) AS cpu_velocity  
FROM TABLE(MON_SAMPLE_SERVICE_CLASS_METRICS  
            (null, current server, '', '', 30, -2)) AS t  
WHERE service_subclass_name IS NOT NULL  
GROUP BY service_superclass_name, service_subclass_name  
ORDER BY service_superclass_name, service_subclass_name
```

This an example of output from this query.

| SERVICE_SUPERCLASS | SERVICE_SUBCLASS | ... |
|----------------------------|--------------------|-----|
| ----- | ----- | ... |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | ... |
| SYSDEFAULTMAINTENANCECLASS | SYSDEFAULTSUBCLASS | ... |
| SYSDEFAULTSYSTEMCLASS | SYSDEFAULTSUBCLASS | ... |

3 record(s) selected.

Output for query (continued).

| ... | ACT_THROUGHPUT | CPU_VELOCITY |
|-----|----------------|--------------|
| ... | ----- | ----- |
| ... | 214.76 | 0.72 |
| ... | 0 | 0 |
| ... | 0 | 0 |

Example 2

Show the configured shares, the estimated CPU entitlement, and the actual CPU utilization for a 5 minute period, for each service class across all databases and partitions on the system.

```

SELECT varchar(db_name,18) AS db_name,
       varchar(service_superclass_name,30) AS service_superclass,
       varchar(service_subclass_name,30) AS service_subclass,
       cpu_shares,
       cpu_limit,
       decimal(estimated_cpu_entitlement, 9, 2) AS estimated_cpu_entitlement,
       decimal(cpu_utilization, 3, 2) AS cpu_utilization
FROM TABLE(MON_SAMPLE_SERVICE_CLASS_METRICS
            (null, null, '', 'T', 300, -2)) AS t
ORDER BY db_name, service_superclass_name, service_subclass_name, member

```

This an example of output from this query.

| DB_NAME | SERVICE_SUPERCLASS | SERVICE_SUBCLASS | ... |
|---------|----------------------------|--------------------|-----|
| SAMPLE | SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | ... |
| SAMPLE | SYSDEFAULTUSERCLASS | - | ... |
| SAMPLE | SYSDEFAULTMAINTENANCECLASS | SYSDEFAULTSUBCLASS | ... |
| SAMPLE | SYSDEFAULTMAINTENANCECLASS | - | ... |
| SAMPLE | SYSDEFAULTSYSTEMCLASS | SYSDEFAULTSUBCLASS | ... |
| SAMPLE | SYSDEFAULTSYSTEMCLASS | - | ... |
| OTHER | SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | ... |
| OTHER | SYSDEFAULTUSERCLASS | - | ... |
| OTHER | SYSDEFAULTMAINTENANCECLASS | SYSDEFAULTSUBCLASS | ... |
| OTHER | SYSDEFAULTMAINTENANCECLASS | - | ... |
| OTHER | SYSDEFAULTSYSTEMCLASS | SYSDEFAULTSUBCLASS | ... |
| OTHER | SYSDEFAULTSYSTEMCLASS | - | ... |

12 record(s) selected.

Output for query (continued).

| ... | CPU_SHARES | CPU_LIMIT | ESTIMATED_CPU_ENTITLEMENT | CPU_UTILIZATION |
|-----|------------|-----------|---------------------------|-----------------|
| ... | 1000 | - | 0.20 | 0.16 |
| ... | 2000 | 70 | 0.20 | 0.16 |
| ... | 1000 | - | 0.10 | 0.09 |
| ... | 1000 | - | 0.10 | 0.09 |
| ... | - | - | - | 0.10 |
| ... | - | - | - | 0.10 |
| ... | 1000 | - | 0.50 | 0.45 |
| ... | 5000 | 70 | 0.50 | 0.45 |
| ... | 1000 | - | 0.20 | 0.11 |
| ... | 2000 | - | 0.20 | 0.11 |
| ... | - | - | - | 0.09 |
| ... | - | - | - | 0.09 |

MON_SAMPLE_WORKLOAD_METRICS - Get sample

The MON_SAMPLE_WORKLOAD_METRICS table function reads system metrics for one or more workloads across one or more databases at two points in time: at the time the function is called and after a given amount of time has passed.

The function computes various statistics from these metrics.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

None

Syntax

```
►►—MON_SAMPLE_SERVICE_CLASS_METRICS—(—hostname—,—db_name—,—  
►workload_name—,—sample_time—,—member—)—►►
```

The schema is SYSPROC.

Table function parameters

hostname

An input argument of type VARCHAR(255) that specifies a fully qualified host name or an IP address of the node from which to generate the report. If the argument is NULL or an empty string, metrics are taken from all nodes in the instance.

db_name

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database when calling this function. The database must have a directory entry type of either "Indirect" or "Home", as returned by a LIST DATABASE DIRECTORY command. If the argument is NULL or an empty string, metrics are taken from all active databases in the instance.

workload_name

An input argument of type VARCHAR(128) that specifies a specific workload for which the metrics are to be returned. If the argument is NULL or an empty string, metrics are returned for all workloads in the instance whose database name satisfies the constraint imposed by the db_name parameter.

sample_time

The amount of time the function collects data before computing a result on that data. This value is measured in seconds and must be greater than or equal to 1.

member

An input argument of type INTEGER that specifies a valid member in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all database members. If the NULL value is specified, -1 is set implicitly.

Information returned

Table 179. Information returned from MON_SAMPLE_WORKLOAD_METRICS

| Column name | Data type | Description or corresponding monitor element |
|---------------|--------------|---|
| HOSTNAME | VARCHAR(255) | hostname - Host name monitor element |
| DB_NAME | VARCHAR(128) | db_name - Database name monitor element |
| WORKLOAD_NAME | VARCHAR(128) | workload_name - Workload name monitor element |

Table 179. Information returned from MON_SAMPLE_WORKLOAD_METRICS (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-----------|---|
| WORKLOAD_ID | INTEGER | workload_id - Workload ID monitor element |
| MEMBER | SMALLINT | member - Database member |
| UOW_THROUGHPUT | DOUBLE | uow_throughput - Unit of work throughput monitor element |
| UOW_LIFETIME_AVG | DOUBLE | uow_lifetime_avg - Unit of work lifetime average monitor element |
| UOW_COMPLETED_TOTAL | BIGINT | uow_completed_total - Total completed units of work monitor element |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time monitor element |
| TOTAL_DISP_RUN_QUEUE_TIME | BIGINT | total_disp_run_queue_time - Total dispatcher run queue time monitor element |
| ACT_THROUGHPUT | DOUBLE | act_throughput - Activity throughput monitor element |
| ACT_COMPLETED_TOTAL | BIGINT | act_completed_total - Total completed activities |
| CPU_UTILIZATION | DOUBLE | cpu_utilization - CPU utilization monitor element |
| CPU_VELOCITY | DOUBLE | cpu_velocity - CPU velocity monitor element |

Usage notes

The MON_SAMPLE_WORKLOAD_METRICS table function returns one row of data per workload and per member for each database. The table function performs no aggregation across workloads (on a member), or across members (for a workload or more). However, aggregation can be achieved through SQL queries.

Example

Display the unit of work (UOW) throughput, activity throughput, and average CPU utilization for a 30 second period, for each workload and across all partitions.

```
SELECT varchar(workload_name,30) AS workload_name,
       decimal(sum(uow_throughput),10,2) AS uow_throughput,
       decimal(sum(act_throughput),10,2) AS act_throughput,
       decimal(sum(cpu_utilization),3,2) AS cpu_utilization
FROM TABLE(MON_SAMPLE_WORKLOAD_METRICS
            (null, current server, '',30 ,-2)) AS t
GROUP BY workload_name
ORDER BY workload_name
```

This an example of output from this query.

```
WORKLOAD_NAME          UOW_THROUGHPUT          ...
-----
SYSDEFAULTUSERWORKLOAD          124.43 ...
SYSDEFAULTADMWORKLOAD          0 ...
```

2 record(s) selected.

Output for query (continued).

| ... | ACT_THROUGHPUT | CPU_UTILIZATION |
|-----|----------------|-----------------|
| ... | 214.76 | 0.89 |
| ... | 0 | 0 |

WebSphere MQ routines

This grouping of routines and views can be used to interact with messages in an IBM WebSphere® MQ (formerly known as IBM MQSeries®) environment.

MQPUBLISH

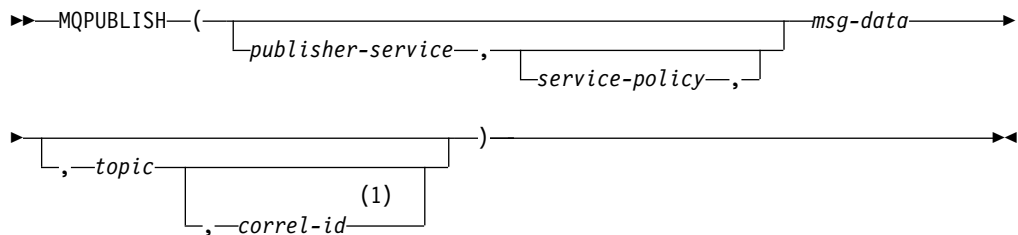
The MQPUBLISH function publishes data to MQSeries.

For more details, visit <http://www.ibm.com/software/mqseries/>.

The MQPUBLISH function publishes the data contained in *msg-data* to the MQSeries publisher specified in *publisher-service*, and using the quality of service policy defined by *service-policy*. An optional topic for the message can be specified, and an optional user-defined message correlation identifier can also be specified.

The data type of the result is VARCHAR(1). The result of the function is '1' if successful or '0' if unsuccessful.

Syntax



Notes:

- 1 The *correl-id* cannot be specified unless a *service* and a *policy* are also specified.

The schema is DB2MQ for non-transactional message queuing functions, and DB2MQ1C for one-phase commit transactional MQ functions.

Function parameters

publisher-service

A string containing the logical MQSeries destination where the message is to be sent. If specified, the *publisher-service* must refer to a publisher Service Point defined in the DB2MQ.MQPUBSUB table that has a type value of 'P' for publisher service. If *publisher-service* is not specified, the DB2.DEFAULT.PUBLISHER will be used. The maximum size of *publisher-service* is 48 bytes.

service-policy

A string containing the MQSeries Service Policy to be used in handling of this message. If specified, the *service-policy* must refer to a Policy defined in the DB2MQ.MQPOLICY table. A Service Policy defines a set of quality of service options that should be applied to this messaging operation. These options

include message priority and message persistence. If *service-policy* is not specified, the default DB2.DEFAULT.POLICY will be used. The maximum size of *service-policy* is 48 bytes.

msg-data

A string expression containing the data to be sent via MQSeries. The maximum size for a VARCHAR string expression is 32 000 bytes and the maximum size for a CLOB string expression is 1M bytes.

topic

A string expression containing the topic for the message publication. If no topic is specified, none will be associated with the message. The maximum size of *topic* is 40 bytes. Multiple topics can be specified in one string (up to 40 characters long). Each topic must be separated by a colon. For example, "t1:t2:the third topic" indicates that the message is associated with all three topics: t1, t2, and "the third topic".

correl-id

An optional string expression containing a correlation identifier to be associated with this message. The *correl-id* is often specified in request and reply scenarios to associate requests with replies. If not specified, no correlation ID will be added to the message. The maximum size of *correl-id* is 24 bytes.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: This example publishes the string "Testing 123" to the default publisher service (DB2.DEFAULT.PUBLISHER) using the default policy (DB2.DEFAULT.POLICY). No correlation identifier or topic is specified for the message.

```
VALUES MQPUBLISH('Testing 123')
```

Example 2: This example publishes the string "Testing 345" to the publisher service "MYPUBLISHER" under the topic "TESTS". The default policy is used and no correlation identifier is specified.

```
VALUES MQPUBLISH('MYPUBLISHER','Testing 345', 'TESTS')
```

Example 3: This example publishes the string "Testing 678" to the publisher service "MYPUBLISHER" using the policy "MYPOLICY" with a correlation identifier of "TEST1". The message is published with topic "TESTS".

```
VALUES MQPUBLISH('MYPUBLISHER','MYPOLICY','Testing 678','TESTS','TEST1')
```

Example 4: This example publishes the string "Testing 901" to the publisher service "MYPUBLISHER" under the topic "TESTS" using the default policy (DB2.DEFAULT.POLICY) and no correlation identifier.

```
VALUES MQPUBLISH('Testing 901','TESTS')
```

MQREAD

The MQREAD function returns a message from the MQSeries location specified by *receive-service*, using the quality of service policy defined in *service-policy*.

Executing this operation does not remove the message from the queue associated with *receive-service*, but instead returns the message at the head of the queue.

The data type of the result is VARCHAR (32000). If no messages are available to be returned, the result is the null value.

Syntax

```
MQREAD ( ( receive-service , service-policy ) )
```

The schema is DB2MQ for non-transactional message queuing functions, and DB2MQ1C for one-phase commit transactional MQ functions.

Function parameters

receive-service

A string containing the logical MQSeries destination from where the message is to be received. If specified, the *receive-service* must refer to a Service Point defined in the DB2MQ.MQSERVICE table. A service point is a logical end-point from where a message is sent or received. Service points definitions include the name of the MQSeries Queue Manager and Queue. If *receive-service* is not specified, then the DB2.DEFAULT.SERVICE will be used. The maximum size of *receive-service* is 48 bytes.

service-policy

A string containing the MQSeries Service Policy used in handling this message. If specified, the *service-policy* must refer to a Policy defined in the DB2MQ.MQPOLICY table. A Service Policy defines a set of quality of service options that should be applied to this messaging operation. These options include message priority and message persistence. If *service-policy* is not specified, then the default DB2.DEFAULT.POLICY will be used. The maximum size of *service-policy* is 48 bytes.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: This example reads the message at the head of the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY).

```
VALUES MQREAD()
```

Example 2: This example reads the message at the head of the queue specified by the service "MYSERVICE" using the default policy (DB2.DEFAULT.POLICY).

```
VALUES MQREAD('MYSERVICE')
```

Example 3: This example reads the message at the head of the queue specified by the service "MYSERVICE", and using the policy "MYPOLICY".

```
VALUES MQREAD('MYSERVICE', 'MYPOLICY')
```

MQREADALL

The MQREADALL table function returns a table containing the messages and message metadata from the MQSeries location specified by *receive-service*, using the quality of service policy *service-policy*.

Performing this operation does not remove the messages from the queue associated with *receive-service*.

Syntax

```
➤➤ MQREADALL ( ( receive-service , service-policy ) ) ➤➤  
               num-rows
```

The schema is DB2MQ for non-transactional message queuing functions, and DB2MQ1C for one-phase commit transactional MQ functions.

Table function parameters

receive-service

A string containing the logical MQSeries destination from which the message is read. If specified, the *receive-service* must refer to a service point defined in the DB2MQ.MQSERVICE table. A service point is a logical end-point from which a message is sent or received. Service point definitions include the name of the MQSeries Queue Manager and Queue. If *receive-service* is not specified, then the DB2.DEFAULT.SERVICE will be used. The maximum size of *receive-service* is 48 bytes.

service-policy

A string containing the MQSeries Service Policy used in the handling of this message. If specified, the *service-policy* refers to a Policy defined in the DB2MQ.MQPOLICY table. A service policy defines a set of quality of service options that should be applied to this messaging operation. These options include message priority and message persistence. If *service-policy* is not specified, then the default DB2.DEFAULT.POLICY will be used. The maximum size of *service-policy* is 48 bytes.

num-rows

A positive integer containing the maximum number of messages to be returned by the function.

If *num-rows* is specified, then a maximum of *num-rows* messages will be returned. If *num-rows* is not specified, then all available messages will be returned.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: This example receives all the messages from the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY). The messages and all the metadata are returned as a table.

```
SELECT * FROM table (MQREADALL()) AS T
```

Example 2: This example receives all the messages from the head of the queue specified by the service MYSERVICE, using the default policy (DB2.DEFAULT.POLICY). Only the MSG and CORRELID columns are returned.

```
SELECT T.MSG, T.CORRELID FROM table (MQREADALL('MYSERVICE')) AS T
```

Example 3: This example reads the head of the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY). Only messages with a CORRELID of '1234' are returned. All columns are returned.

```
SELECT * FROM table (MQREADALL()) AS T WHERE T.CORRELID = '1234'
```

Example 4: This example receives the first 10 messages from the head of the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY). All columns are returned.

```
SELECT * FROM table (MQREADALL(10)) AS T
```

Information returned

Table 180. Information returned by the MQREADALL table function

| Column name | Data type | Description |
|-------------|----------------|--|
| MSG | VARCHAR(32000) | Contains the contents of the MQSeries message. |

Table 180. Information returned by the MQREADALL table function (continued)

| Column name | Data type | Description |
|-------------|-------------|---|
| CORRELID | VARCHAR(24) | Contains a correlation ID that can be used to identify messages. You can select a message from the queue using this identifier. In the case of a request and response scenario, the correlation ID enables you to associate a response with a particular request. |
| TOPIC | VARCHAR(40) | Contains the topic with which the message was published, if available. |
| QNAME | VARCHAR(48) | Contains the name of the queue where the message was received. |
| MSGID | CHAR(24) | Contains the assigned unique MQSeries identifier for this message. |
| MSGFORMAT | VARCHAR(8) | Contains the format of the message, as defined by MQSeries. Typical strings have an MQSTR format. |

MQREADALLCLOB

The MQREADALLCLOB table function returns a table containing the messages and message metadata from the MQSeries location specified by *receive-service*, using the quality of service policy *service-policy*.

Performing this operation does not remove the messages from the queue associated with *receive-service*.

Syntax

```

MQREADALLCLOB
(
  ( receive-service )
  ( ,—service-policy )
  ( num-rows )
)

```

The schema is DB2MQ.

Table function parameters

receive-service

A string containing the logical MQSeries destination from which the message is read. If specified, the *receive-service* must refer to a service point defined in the DB2MQ.MQSERVICE table. A service point is a logical end-point from which a message is sent or received. Service point definitions include the name of the MQSeries Queue Manager and Queue. If *receive-service* is not specified, then the DB2.DEFAULT.SERVICE will be used. The maximum size of *receive-service* is 48 bytes.

service-policy

A string containing the MQSeries Service Policy used in the handling of this message. If specified, the *service-policy* refers to a Policy defined in the DB2MQ.MQPOLICY table. A service policy defines a set of quality of service options that should be applied to this messaging operation. These options include message priority and message persistence. If *service-policy* is not specified, then the default DB2.DEFAULT.POLICY will be used. The maximum size of *service-policy* is 48 bytes.

num-rows

A positive integer containing the maximum number of messages to be returned by the function.

If *num-rows* is specified, then a maximum of *num-rows* messages will be returned. If *num-rows* is not specified, then all available messages will be returned.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: This example receives all the messages from the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY). The messages and all the metadata are returned as a table.

```
SELECT * FROM table (MQREADALLCLOB()) AS T
```

Example 2: This example receives all the messages from the head of the queue specified by the service MYSERVICE, using the default policy (DB2.DEFAULT.POLICY). Only the MSG and CORRELID columns are returned.

```
SELECT T.MSG, T.CORRELID FROM table (MQREADALLCLOB('MYSERVICE')) AS T
```

Example 3: This example reads the head of the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY). Only messages with a CORRELID of '1234' are returned. All columns are returned.

```
SELECT * FROM table (MQREADALLCLOB()) AS T WHERE T.CORRELID = '1234'
```

Example 4: This example receives the first 10 messages from the head of the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY). All columns are returned.

```
SELECT * FROM table (MQREADALLCLOB(10)) AS T
```

Information returned

Table 181. Information returned by the MQREADALLCLOB table function

| Column name | Data type | Description |
|-------------|-------------|---|
| MSG | CLOB(1M) | Contains the contents of the MQSeries message. |
| CORRELID | VARCHAR(24) | Contains a correlation ID that can be used to identify messages. You can select a message from the queue using this identifier. In the case of a request and response scenario, the correlation ID enables you to associate a response with a particular request. |
| TOPIC | VARCHAR(40) | Contains the topic with which the message was published, if available. |
| QNAME | VARCHAR(48) | Contains the name of the queue where the message was received. |
| MSGID | CHAR(24) | Contains the assigned unique MQSeries identifier for this message. |
| MSGFORMAT | VARCHAR(8) | Contains the format of the message, as defined by MQSeries. Typical strings have an MQSTR format. |

MQREADCLOB

The MQREADCLOB function returns a message from the MQSeries location specified by *receive-service*, using the quality of service policy defined in *service-policy*.

Executing this operation does not remove the message from the queue associated with *receive-service*, but instead returns the message at the head of the queue.

The data type of the result is CLOB(1M). If no messages are available to be returned, the result is the null value.

Syntax

►► MQREADCLOB ((*receive-service* [, *service-policy*])) ►►

The schema is DB2MQ.

Function parameters

receive-service

A string containing the logical MQSeries destination from where the message is to be received. If specified, the *receive-service* must refer to a Service Point

defined in the DB2MQ.MQSERVICE table. A service point is a logical end-point from where a message is sent or received. Service points definitions include the name of the MQSeries Queue Manager and Queue. If *receive-service* is not specified, then the DB2.DEFAULT.SERVICE will be used. The maximum size of *receive-service* is 48 bytes.

service-policy

A string containing the MQSeries Service Policy used in handling this message. If specified, the *service-policy* must refer to a Policy defined in the DB2MQ.MQPOLICY table. A Service Policy defines a set of quality of service options that should be applied to this messaging operation. These options include message priority and message persistence. If *service-policy* is not specified, then the default DB2.DEFAULT.POLICY will be used. The maximum size of *service-policy* is 48 bytes.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: This example reads the message at the head of the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY).

```
VALUES MQREADCLOB()
```

Example 2: This example reads the message at the head of the queue specified by the service "MYSERVICE" using the default policy (DB2.DEFAULT.POLICY).

```
VALUES MQREADCLOB('MYSERVICE')
```

Example 3: This example reads the message at the head of the queue specified by the service "MYSERVICE", and using the policy "MYPOLICY".

```
VALUES MQREADCLOB('MYSERVICE', 'MYPOLICY')
```

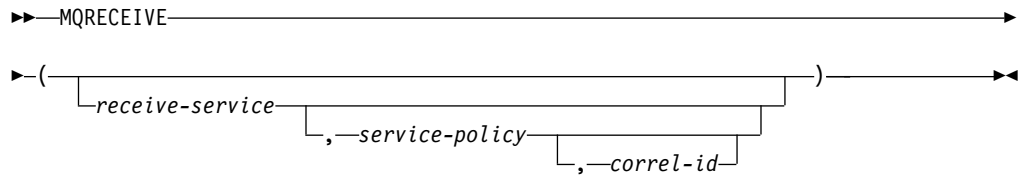
MQRECEIVE

The MQRECEIVE function returns a message from the MQSeries location specified by *receive-service*, using the quality of service policy *service-policy*.

Performing this operation removes the message from the queue associated with *receive-service*. If the *correl-id* is specified, then the first message with a matching correlation identifier will be returned. If *correl-id* is not specified, then the message at the head of the queue will be returned.

The data type of the result is VARCHAR (32000). If no messages are available to be returned, the result is the null value.

Syntax



The schema is DB2MQ for non-transactional message queuing functions, and DB2MQ1C for one-phase commit transactional MQ functions.

Function parameters

receive-service

A string containing the logical MQSeries destination from which the message is received. If specified, the *receive-service* must refer to a Service Point defined in the DB2MQ.MQSERVICE table. A service point is a logical end-point from which a message is sent or received. Service points definitions include the name of the MQSeries Queue Manager and Queue. If *receive-service* is not specified, the DB2.DEFAULT.SERVICE is used. The maximum size of *receive-service* is 48 bytes.

service-policy

A string containing the MQSeries Service Policy to be used in the handling of this message. If specified, *service-policy* must refer to a policy defined in the DB2MQ.MQPOLICY table. A service policy defines a set of quality of service options that should be applied to this messaging operation. These options include message priority and message persistence. If *service-policy* is not specified, the default DB2.DEFAULT.POLICY is used. The maximum size of *service-policy* is 48 bytes.

correl-id

A string containing an optional correlation identifier to be associated with this message. The *correl-id* is often specified in request and reply scenarios to associate requests with replies. If not specified, no correlation id will be specified. The maximum size of *correl-id* is 24 bytes.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: This example receives the message at the head of the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY).

VALUES MQRECEIVE()

Example 2: This example receives the message at the head of the queue specified by the service "MYSERVICE" using the default policy (DB2.DEFAULT.POLICY).

```
VALUES MQRECEIVE('MYSERVICE')
```

Example 3: This example receives the message at the head of the queue specified by the service "MYSERVICE" using the policy "MYPOLICY".

```
VALUES MQRECEIVE('MYSERVICE', 'MYPOLICY')
```

Example 4: This example receives the first message with a correlation id that matches '1234' from the head of the queue specified by the service "MYSERVICE" using the policy "MYPOLICY".

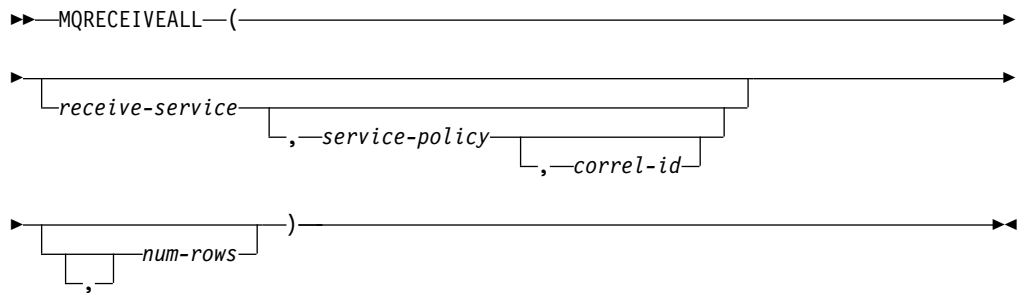
```
VALUES MQRECEIVE('MYSERVICE', 'MYPOLICY', '1234')
```

MQRECEIVEALL

The MQRECEIVEALL table function returns a table containing the messages and message metadata from the MQSeries location specified by *receive-service*, using the quality of service policy *service-policy*.

Performing this operation removes the messages from the queue associated with *receive-service*.

Syntax



The schema is DB2MQ for non-transactional message queuing functions, and DB2MQ1C for one-phase commit transactional MQ functions.

Table function parameters

receive-service

A string containing the logical MQSeries destination from which the message is received. If specified, the *receive-service* must refer to a service point defined in the DB2MQ.MQSERVICE table. A service point is a logical end-point from which a message is sent or received. Service point definitions include the name of the MQSeries Queue Manager and Queue. If *receive-service* is not specified, then the DB2.DEFAULT.SERVICE will be used. The maximum size of *receive-service* is 48 bytes.

service-policy

A string containing the MQSeries Service Policy used in the handling of this message. If specified, the *service-policy* refers to a Policy defined in the DB2MQ.MQPOLICY table. A service policy defines a set of quality of service options that should be applied to this messaging operation. These options

include message priority and message persistence. If *service-policy* is not specified, then the default DB2.DEFAULT.POLICY will be used. The maximum size of *service-policy* is 48 bytes.

correl-id

An optional string containing a correlation identifier associated with this message. The *correl-id* is often specified in request and reply scenarios to associate requests with replies. If not specified, no correlation id is specified. The maximum size of *correl-id* is 24 bytes.

If a *correl-id* is specified, all the messages with a matching correlation identifier are returned and removed from the queue. If *correl-id* is not specified, the message at the head of the queue is returned.

num-rows

A positive integer containing the maximum number of messages to be returned by the function.

If *num-rows* is specified, then a maximum of *num-rows* messages will be returned. If *num-rows* is not specified, then all available messages will be returned.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: This example receives all the messages from the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY). The messages and all the metadata are returned as a table.

```
SELECT * FROM table (MQRECEIVEALL()) AS T
```

Example 2: This example receives all the messages from the head of the queue specified by the service MYSERVICE, using the default policy (DB2.DEFAULT.POLICY). Only the MSG and CORRELID columns are returned.

```
SELECT T.MSG, T.CORRELID FROM table (MQRECEIVEALL('MYSERVICE')) AS T
```

Example 3: This example receives all of the message from the head of the queue specified by the service "MYSERVICE", using the policy "MYPOLICY". Only messages with a CORRELID of '1234' are returned. Only the MSG and CORRELID columns are returned.

```
SELECT T.MSG, T.CORRELID FROM table  
(MQRECEIVEALL('MYSERVICE', 'MYPOLICY', '1234')) AS T
```

Example 4: This example receives the first 10 messages from the head of the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY). All columns are returned.

```
SELECT * FROM table (MQRECEIVEALL(10)) AS T
```

Information returned

Table 182. Information returned by the MQRECEIVEALL table function

| Column name | Data type | Description |
|-------------|----------------|---|
| MSG | VARCHAR(32000) | Contains the contents of the MQSeries message. |
| CORRELID | VARCHAR(24) | Contains a correlation ID that can be used to identify messages. You can select a message from the queue using this identifier. In the case of a request and response scenario, the correlation ID enables you to associate a response with a particular request. |
| TOPIC | VARCHAR(40) | Contains the topic with which the message was published, if available. |
| QNAME | VARCHAR(48) | Contains the name of the queue where the message was received. |
| MSGID | CHAR(24) | Contains the assigned unique MQSeries identifier for this message. |
| MSGFORMAT | VARCHAR(8) | Contains the format of the message, as defined by MQSeries. Typical strings have an MQSTR format. |

MQRECEIVEALLCLOB

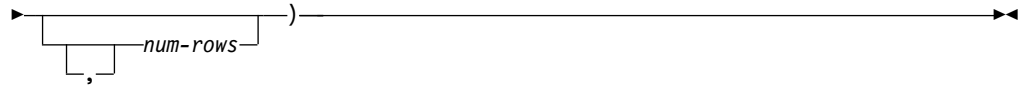
The MQRECEIVEALLCLOB table function returns a table containing the messages and message metadata from the MQSeries location specified by *receive-service*, using the quality of service policy *service-policy*.

Performing this operation removes the messages from the queue associated with *receive-service*.

Syntax

```

MQRECEIVEALLCLOB—(
  receive-service
  ,—service-policy
  ,—correl-id
)
```



The schema is DB2MQ.

Table function parameters

receive-service

A string containing the logical MQSeries destination from which the message is received. If specified, the *receive-service* must refer to a service point defined in the DB2MQ.MQSERVICE table. A service point is a logical end-point from which a message is sent or received. Service point definitions include the name of the MQSeries Queue Manager and Queue. If *receive-service* is not specified, then the DB2.DEFAULT.SERVICE will be used. The maximum size of *receive-service* is 48 bytes.

service-policy

A string containing the MQSeries Service Policy used in the handling of this message. If specified, the *service-policy* refers to a Policy defined in the DB2MQ.MQPOLICY table. A service policy defines a set of quality of service options that should be applied to this messaging operation. These options include message priority and message persistence. If *service-policy* is not specified, then the default DB2.DEFAULT.POLICY will be used. The maximum size of *service-policy* is 48 bytes.

correl-id

An optional string containing a correlation identifier associated with this message. The *correl-id* is often specified in request and reply scenarios to associate requests with replies. If not specified, no correlation id is specified. The maximum size of *correl-id* is 24 bytes.

If a *correl-id* is specified, then only those messages with a matching correlation identifier will be returned. If *correl-id* is not specified, then the message at the head of the queue will be returned.

num-rows

A positive integer containing the maximum number of messages to be returned by the function.

If *num-rows* is specified, then a maximum of *num-rows* messages will be returned. If *num-rows* is not specified, then all available messages are returned.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: This example receives all the messages from the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY). The messages and all the metadata are returned as a table.

```
SELECT * FROM table (MQRECEIVEALLCLOB()) AS T
```

Example 2: This example receives all the messages from the head of the queue specified by the service MYSERVICE, using the default policy (DB2.DEFAULT.POLICY). Only the MSG and CORRELID columns are returned.

```
SELECT T.MSG, T.CORRELID  
FROM table (MQRECEIVEALLCLOB('MYSERVICE')) AS T
```

Example 3: This example receives all of the message from the head of the queue specified by the service "MYSERVICE", using the policy "MYPOLICY". Only messages with a CORRELID of '1234' are returned. Only the MSG and CORRELID columns are returned.

```
SELECT T.MSG, T.CORRELID  
FROM table (MQRECEIVEALLCLOB('MYSERVICE','MYPOLICY','1234')) AS T
```

Example 4: This example receives the first 10 messages from the head of the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY). All columns are returned.

```
SELECT * FROM table (MQRECEIVEALLCLOB(10)) AS T
```

Information returned

Table 183. Information returned by the MQRECEIVEALLCLOB table function

| Column name | Data type | Description |
|-------------|-------------|---|
| MSG | CLOB(1M) | Contains the contents of the MQSeries message. |
| CORRELID | VARCHAR(24) | Contains a correlation ID that can be used to identify messages. You can select a message from the queue using this identifier. In the case of a request and response scenario, the correlation ID enables you to associate a response with a particular request. |
| TOPIC | VARCHAR(40) | Contains the topic with which the message was published, if available. |
| QNAME | VARCHAR(48) | Contains the name of the queue where the message was received. |
| MSGID | CHAR(24) | Contains the assigned unique MQSeries identifier for this message. |
| MSGFORMAT | VARCHAR(8) | Contains the format of the message, as defined by MQSeries. Typical strings have an MQSTR format. |

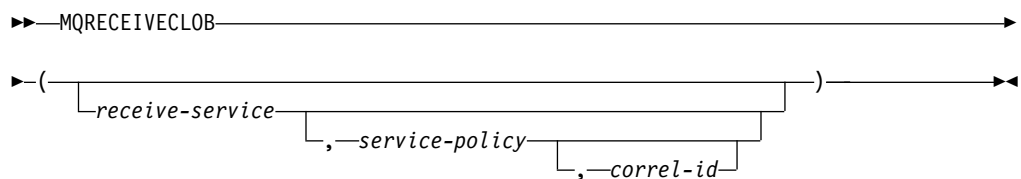
MQRECEIVECLOB

The MQRECEIVECLOB function returns a message from the MQSeries location specified by *receive-service*, using the quality of service policy *service-policy*.

Performing this operation removes the message from the queue associated with *receive-service*. If the *correl-id* is specified, the first message with a matching correlation identifier will be returned. If *correl-id* is not specified, the message at the head of the queue will be returned.

The data type of the result is CLOB(1M). If no messages are available to be returned, the result is the null value.

Syntax



The schema is DB2MQ.

Function parameters

receive-service

A string containing the logical MQSeries destination from which the message is received. If specified, the *receive-service* must refer to a Service Point defined in the DB2MQ.MQSERVICE table. A service point is a logical end-point from which a message is sent or received. Service points definitions include the name of the MQSeries Queue Manager and Queue. If *receive-service* is not specified, the DB2.DEFAULT.SERVICE is used. The maximum size of *receive-service* is 48 bytes.

service-policy

A string containing the MQSeries Service Policy to be used in the handling of this message. If specified, the *service-policy* must refer to a policy defined in the DB2MQ.MQPOLICY table. A service policy defines a set of quality of service options that should be applied to this messaging operation. These options include message priority and message persistence. If *service-policy* is not specified, the default DB2.DEFAULT.POLICY is used. The maximum size of *service-policy* is 48 bytes.

correl-id

A string containing an optional correlation identifier to be associated with this message. The *correl-id* is often specified in request and reply scenarios to associate requests with replies. If not specified, no correlation id will be used. The maximum size of *correl-id* is 24 bytes.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority

- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: This example receives the message at the head of the queue specified by the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY).

```
VALUES MQRECEIVECLOB()
```

Example 2: This example receives the message at the head of the queue specified by the service "MYSERVICE" using the default policy (DB2.DEFAULT.POLICY).

```
VALUES MQRECEIVECLOB('MYSERVICE')
```

Example 3: This example receives the message at the head of the queue specified by the service "MYSERVICE" using the policy "MYPOLICY".

```
VALUES MQRECEIVECLOB('MYSERVICE', 'MYPOLICY')
```

Example 4: This example receives the first message with a correlation ID that matches '1234' from the head of the queue specified by the service "MYSERVICE" using the policy "MYPOLICY".

```
VALUES MQRECEIVECLOB('MYSERVICE', 'MYPOLICY', '1234')
```

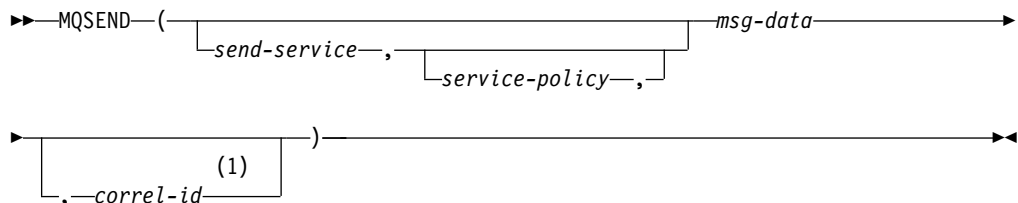
MQSEND

The MQSEND function sends the data contained in *msg-data* to the MQSeries location specified by *send-service*, using the quality of service policy defined by *service-policy*.

An optional user-defined message correlation identifier can be specified using *correl-id*.

The data type of the result is VARCHAR(1). The result of the function is '1' if successful or '0' if unsuccessful.

Syntax



Notes:

- 1 The *correl-id* cannot be specified unless a *service* and a *policy* are also specified.

The schema is DB2MQ for non-transactional message queuing functions, and DB2MQ1C for one-phase commit transactional MQ functions.

Function parameters

msg-data

A string expression containing the data to be sent via MQSeries. The maximum size for a VARCHAR string expression is 32 000 bytes and the maximum size for a CLOB string expression is 1M bytes.

send-service

A string containing the logical MQSeries destination where the message is to be sent. If specified, the *send-service* refers to a service point defined in the DB2MQ.MQSERVICE table. A service point is a logical end-point from which a message may be sent or received. Service point definitions include the name of the MQSeries Queue Manager and Queue. If *send-service* is not specified, the value of DB2.DEFAULT.SERVICE is used. The maximum size of *send-service* is 48 bytes.

service-policy

A string containing the MQSeries Service Policy used in handling of this message. If specified, the *service-policy* must refer to a service policy defined in the DB2MQ.MQPOLICY table. A Service Policy defines a set of quality of service options that should be applied to this messaging operation. These options include message priority and message persistence. If *service-policy* is not specified, a default value of DB2.DEFAULT.POLICY will be used. The maximum size of *service-policy* is 48 bytes.

correl-id

An optional string containing a correlation identifier associated with this message. The *correl-id* is often specified in request and reply scenarios to associate requests with replies. If not specified, no correlation ID will be sent. The maximum size of *correl-id* is 24 bytes.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: This example sends the string "Testing 123" to the default service (DB2.DEFAULT.SERVICE), using the default policy (DB2.DEFAULT.POLICY), with no correlation identifier.

```
VALUES MQSEND('Testing 123')
```

Example 2: This example sends the string "Testing 345" to the service "MYSERVICE", using the policy "MYPOLICY", with no correlation identifier.

```
VALUES MQSEND('MYSERVICE', 'MYPOLICY', 'Testing 345')
```

Example 3: This example sends the string "Testing 678" to the service "MYSERVICE", using the policy "MYPOLICY", with correlation identifier "TEST3".

```
VALUES MQSEND('MYSERVICE','MYPOLICY','Testing 678','TEST3')
```

Example 4: This example sends the string "Testing 901" to the service "MYSERVICE", using the default policy (DB2.DEFAULT.POLICY), and no correlation identifier.

```
VALUES MQSEND('MYSERVICE','Testing 901')
```

MQSUBSCRIBE

The MQSUBSCRIBE function is used to register interest in MQSeries messages published on a specified topic.

Successful execution of this function causes the publish and subscribe server to forward messages matching the topic to the service point defined by *subscriber-service*.

The *subscriber-service* specifies a logical destination for messages that match the specified topic. Messages that match *topic* are placed on the queue defined by *subscriber-service*, and can be read or received through a subsequent call to MQREAD, MQRECEIVE, MQREADALL, or MQRECEIVEALL. For more details, visit <http://www.ibm.com/software/mqseries/>.

The data type of the result is VARCHAR(1). The result of the function is '1' if successful or '0' if unsuccessful.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Syntax

```
►► MQSUBSCRIBE ( subscriber-service , service-policy , topic ) ►►
```

The schema is DB2MQ for non-transactional message queuing functions, and DB2MQ1C for one-phase commit transactional MQ functions.

Function parameters

subscriber-service

A string containing the logical MQSeries subscription point to where messages matching *topic* will be sent. If specified, the *subscriber-service* must refer to a Subscribers Service Point defined in the DB2MQ.MQPUBSUB table that has a type value of 'S' for publisher service. If *subscriber-service* is not specified, then the DB2.DEFAULT.SUBSCRIBER will be used instead. The maximum size of *subscriber-service* is 48 bytes.

service-policy

A string containing the MQSeries Service Policy to be used in handling the message. If specified, the *service-policy* must refer to a Policy defined in the DB2MQ.MQPOLICY table. A Service Policy defines a set of quality of service options to be applied to this messaging operation. These options include message priority and message persistence. If *service-policy* is not specified, then the default DB2.DEFAULT.POLICY will be used instead. The maximum size of *service-policy* is 48 bytes.

topic

A string defining the types of messages to receive. Only messages published with the specified topics will be received by this subscription. Multiple subscriptions can coexist. The maximum size of *topic* is 40 bytes. Multiple topics can be specified in one string (up to 40 bytes long). Each topic must be separated by a colon. For example, "t1:t2:the third topic" indicates that the message is associated with all three topics: t1, t2, and "the third topic".

Examples

Example 1: This example registers an interest in messages containing the topic "Weather". The default subscriber-service (DB2.DEFAULT.SUBSCRIBER) is registered as the subscriber and the default service-policy (DB2.DEFAULT.POLICY) specifies the quality of service.

```
VALUES MQSUBSCRIBE('Weather')
```

Example 2: This example demonstrates a subscriber registering interest in messages containing "Stocks". The subscriber registers as "PORTFOLIO-UPDATES" with policy "BASIC-POLICY".

```
VALUES MQSUBSCRIBE('PORTFOLIO-UPDATES', 'BASIC-POLICY', 'Stocks')
```

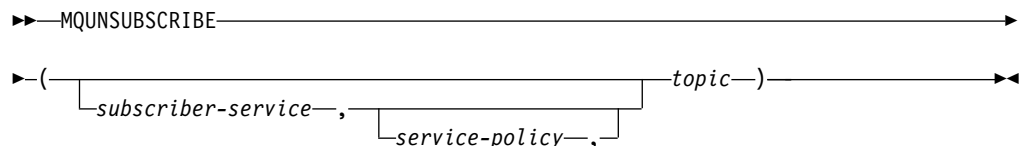
MQUNSUBSCRIBE

The MQUNSUBSCRIBE function is used to unregister an existing message subscription.

The *subscriber-service*, *service-policy*, and *topic* are used to identify the subscription that is to be cancelled. Successful execution of this function causes the publish and subscribe server to remove the specified subscription. Messages with the specified *topic* will no longer be sent to the logical destination defined by *subscriber-service*. For more details, visit <http://www.ibm.com/software/mqseries/>.

The data type of the result is VARCHAR(1). The result of the function is '1' if successful or '0' if unsuccessful.

Syntax



The schema is DB2MQ for non-transactional message queuing functions, and DB2MQ1C for one-phase commit transactional MQ functions.

Function parameters

subscriber-service

If specified, the *subscriber-service* must refer to a Subscribers Service Point defined in the DB2MQ.MQPUBSUB table that has a type value of 'S' for publisher service. If *subscriber-service* is not specified, then the DB2.DEFAULT.SUBSCRIBER will be used instead. The maximum size of subscriber-service is 48 bytes.

service-policy

If specified, the *service-policy* must refer to a Policy defined in the DB2MQ.MQPOLICY table. A Service Policy defines a set of quality of service options to be applied to this messaging operation. If *service-policy* is not specified, then the default DB2.DEFAULT.POLICY will be used. The maximum size of *service-policy* is 48 bytes.

topic

A string specifying the subject of messages that are not to be received. The maximum size of *topic* is 40 bytes. Multiple topics can be specified in one string (up to 40 bytes long). Each topic must be separated by a colon. For example, "t1:t2:the third topic" indicates that the message is associated with all three topics: t1, t2, and "the third topic".

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: This example cancels an interest in messages containing the topic "Weather". The default subscriber-service (DB2.DEFAULT.SUBSCRIBER) is registered as the unsubscriber and the default service-policy (DB2.DEFAULT.POLICY) specifies the quality of service.

```
VALUES MQUNSUBSCRIBE('Weather')
```

Example 2: This example demonstrates a subscriber canceling an interest in messages containing "Stocks". The subscriber is registered as "PORTFOLIO-UPDATES" with policy "BASIC-POLICY".

```
VALUES MQUNSUBSCRIBE('PORTFOLIO-UPDATES', 'BASIC-POLICY', 'Stocks')
```

Security routines and views

This grouping of routines and views can be used to retrieve information about security and authorities, such as roles, authorization IDs, and groups.

ADMIN_ROTATE_MASTER_KEY - Change the database master key

The ADMIN_ROTATE_MASTER_KEY procedure changes the master key for an encrypted database.

Authorization

The following authorization is required:

- EXECUTE privilege on the routine. EXECUTE privilege on this routine can be granted only by the security administrator (SECADM).

Default PUBLIC privilege

None

Syntax

►► ADMIN_ROTATE_MASTER_KEY—(—*label*—)—————►►

The schema is SYSPROC.

Procedure parameters

label

An input and output parameter of type VARCHAR(255) that specifies the label of the new master key to be rotated. The label is associated with a master key at the keystore. If a label is specified, it must be associated with a master key at the keystore. If there is no association, the procedure fails. If this parameter is NULL, a new master key and master key label is generated.

Note:

When the **allow_key_insert_without_keystore_backup** configuration knob is off you cannot use the automatically generated master key.

Information returned

Table 184. Information returned by the ADMIN_ROTATE_MASTER_KEY procedure

| Column name | Data type | Description |
|-------------|--------------|---|
| LABEL | VARCHAR(255) | The name of the new master key that protects the database encryption key. |

Usage notes

Rotation of the master key label should be done according to the organization's security policy. This is similar to changing user passwords at a regular interval, which is also enforced by a security policy. Before performing a master key rotation, you must ensure that the specified label is defined at the keystore and mapped to an existing master key.

This procedure does not re-encrypt any of the data stored on disk.

In a partitioned database environment, ensure that all database partitions are active before calling this procedure.

Examples

Rotate the master key with the specified label.

```
CALL SYSPROC.ADMIN_ROTATE_MASTER_KEY('UserGivenLabel')
```

Value of output parameters

Parameter Name : LABEL

Parameter Value : UserGivenLabel

Return Status = 0

Rotate the master key without a specific label.

```
CALL SYSPROC.ADMIN_ROTATE_MASTER_KEY (NULL)
```

Value of output parameters

Parameter Name : LABEL

Parameter Value : DB2_SYSGEN_instancename_dbname_timestamp

Return Status = 0

AUTH_GET_INSTANCE_AUTHID - Get the instance owner authorization ID

The AUTH_GET_INSTANCE_AUTHID scalar function returns the authorization ID of the instance owner

Syntax

►► AUTH_GET_INSTANCE_AUTHID (—) ◀◀

The schema is SYSPROC.

Authorization

EXECUTE privilege on the AUTH_GET_INSTANCE_AUTHID scalar function.

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Information returned

This function returns a value of type VARCHAR(128) that specifies the authorization ID of the instance owner.

Usage notes

A user who holds SYSADM authority does not have implicit DBADM authority. As a result, applications that run under the instance owner account and that perform operations that are not within the scope of SYSADM authority might experience authorization errors such as SQL1092N, SQL0551N, or SQL0552N.

The **UPGRADE DATABASE** and **RESTORE DATABASE** commands no longer grant DBADM authority to the SYSADM group.

To obtain a list of the authorities held by the instance owner authorization ID:

1. Use the SYSPROC.AUTH_GET_INSTANCE_AUTHID() scalar function to determine the instance owner authorization ID. For example:

```
db2 "VALUES SYSPROC.AUTH_GET_INSTANCE_AUTHID()"
```

This command returns.

```
1
-----
BOB
```

1 record(s) selected.

2. Get a list of the authorities for this authorization ID. For example:

```
SELECT * FROM
  TABLE (SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID ('BOB', 'U') ) AS T
ORDER BY AUTHORITY
```

3. If necessary, grant any missing authorities. For example:

```
GRANT DBADM ON DATABASE TO USER BOB
```

Example

The following example shows how to use the command line processor (CLP) to obtain the authorization ID of the instance owner:

```
db2 "VALUES SYSPROC.AUTH_GET_INSTANCE_AUTHID()"
```

The following is an example of output for this command.

```
1
-----,.,-----
ZURBIE
```

1 record(s) selected.

AUTH_LIST_AUTHORITIES_FOR_AUTHID

The AUTH_LIST_AUTHORITIES_FOR_AUTHID table function returns all authorities held by the authorization ID, either found in the database configuration file or granted to an authorization ID directly or indirectly through a group or a role.

Syntax

```
►►—AUTH_LIST_AUTHORITIES_FOR_AUTHID—(—authid—,—authidtype—)—►►
```

The schema is SYSPROC.

Table function parameters

authid

An input argument of type VARCHAR(128) that specifies the authorization ID being queried. The authorization ID can be a user, group or a role. If *authid* is NULL or an empty string, an empty result table is returned.

authidtype

An input argument of type VARCHAR(1) that specifies the authorization ID type being queried. If *authidtype* does not exist, is NULL or an empty string, an empty result table is returned. Possible values for *authidtype* are:

- G: Group
- R: Role
- U: User

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Information returned

Table 185. The information returned for AUTH_LIST_AUTHORITIES_FOR_AUTHID

| Column Name | Data Type | Description |
|-------------|--------------|---|
| AUTHORITY | VARCHAR(128) | Authority held by the authorization ID |
| D_USER | CHAR(1) | Authority granted directly to the <i>authid</i> , when the <i>authidtype</i> is a user (U). If the <i>authidtype</i> is a group (G) or a role (R), then the value is not applicable (*). <ul style="list-style-type: none"> • N = Not held • Y= Held • * = Not applicable |
| D_GROUP | CHAR(1) | Authority granted directly to the <i>authid</i> when the <i>authidtype</i> is a group (G), or to the group to which the <i>authid</i> belongs when the <i>authidtype</i> is a user (U). If the <i>authidtype</i> is a role (R), then the value is not applicable (*). <ul style="list-style-type: none"> • N = Not held • Y= Held • * = Not applicable |
| D_PUBLIC | CHAR(1) | Authority granted directly to the <i>authid</i> called PUBLIC when the <i>authidtype</i> is a user (U) or a group (G). If the <i>authidtype</i> is a role (R), then the value is not applicable (*). <ul style="list-style-type: none"> • N = Not held • Y= Held • * = Not applicable |
| ROLE_USER | CHAR(1) | Authority granted directly to a role granted to the <i>authid</i> , when the <i>authidtype</i> is a user (U). If the <i>authidtype</i> is a group (G) or a role (R), then the value is not applicable (*). The role could be part of a role hierarchy. <ul style="list-style-type: none"> • N = Not held • Y= Held • * = Not applicable |

Table 185. The information returned for AUTH_LIST_AUTHORITIES_FOR_AUTHID (continued)

| Column Name | Data Type | Description |
|-------------|-----------|--|
| ROLE_GROUP | CHAR(1) | Authority granted directly to a role granted to the <i>authid</i> when the <i>authidtype</i> is a group (G). If the <i>authidtype</i> is a user (U) or a role (R), then the value is not applicable (*). The role could be part of a role hierarchy. <ul style="list-style-type: none"> • N = Not held • Y= Held • * = Not applicable |
| ROLE_PUBLIC | CHAR(1) | Authority granted directly to a role granted to the <i>authid</i> called PUBLIC when the <i>authidtype</i> is a user (U) or a group (G). If the <i>authidtype</i> is a role (R), then the value is not applicable (*). The role could be part of a role hierarchy. <ul style="list-style-type: none"> • N = Not held • Y= Held • * = Not applicable |
| D_ROLE | CHAR(1) | Authority granted to a role or to a role granted to the role. If the <i>authidtype</i> is a user (U) or a group (G), then the value is not applicable (*). The role could be part of a role hierarchy. <ul style="list-style-type: none"> • N = Not held • Y= Held • * = Not applicable |

Example

Consider user ALICE who by default holds BIND, CONNECT, CREATETAB and IMPLICIT_SCHEMA privileges through special group PUBLIC. ALICE is a member of a group ADMIN1 who has the following system authorities: SYSADM, SYSCTRL and SYSMAINT. She is also a member of group ADMIN2 who has DBADM authority. Also, ALICE has been granted DBADM and SECADM database authorities. Role R1 was granted to ALICE. LOAD authority was granted to role R1. Role R2 was granted to group ADMIN1. CREATE_NOT_FENCED_ROUTINE authority was granted to role R2.

Example 1: Retrieve all authorities user ALICE has granted either directly to her or indirectly through a group, PUBLIC or a role.

```
SELECT AUTHORITY, D_USER, D_GROUP, D_PUBLIC, ROLE_USER, ROLE_GROUP, ROLE_PUBLIC, D_ROLE
FROM TABLE (SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID ('ALICE', 'U') ) AS T
ORDER BY AUTHORITY
```

| AUTHORITY | D_USER | D_GROUP | D_PUBLIC | ROLE_USER | ROLE_GROUP | ROLE_PUBLIC | D_ROLE |
|---------------------------|--------|---------|----------|-----------|------------|-------------|--------|
| ACCESSCTRL | N | N | N | N | N | N | * |
| BINDADD | N | N | Y | N | N | N | * |
| CONNECT | N | N | Y | N | N | N | * |
| CREATE_EXTERNAL_ROUTINE | N | N | N | N | N | N | * |
| CREATE_NOT_FENCED_ROUTINE | N | N | N | N | Y | N | * |
| CREATETAB | N | N | Y | N | N | N | * |
| DATAACCESS | N | N | N | N | N | N | * |
| DBADM | Y | Y | N | N | N | N | * |
| EXPLAIN | N | N | N | N | N | N | * |
| IMPLICIT_SCHEMA | N | N | Y | N | N | N | * |
| LOAD | N | N | N | Y | N | N | * |
| QUIESCE_CONNECT | N | N | N | N | N | N | * |
| SECADM | Y | N | N | N | N | N | * |
| SQLADM | N | N | N | N | N | N | * |
| SYSADM | * | Y | * | * | * | * | * |
| SYSCTRL | * | Y | * | * | * | * | * |

| | | | | | | | |
|----------|---|---|---|---|---|---|---|
| SYSMAINT | * | Y | * | * | * | * | * |
| SYSMON | * | N | * | * | * | * | * |
| WLMADM | N | N | N | N | N | N | * |

Example 2: Retrieve all authorities group ADMIN1 has granted either directly to it or indirectly through PUBLIC or a role.

```
SELECT AUTHORITY, D_USER, D_GROUP, D_PUBLIC, ROLE_USER, ROLE_GROUP, ROLE_PUBLIC, D_ROLE
FROM TABLE (SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID ('ADMIN1', 'G')) AS T
ORDER BY AUTHORITY
```

| AUTHORITY | D_USER | D_GROUP | D_PUBLIC | ROLE_USER | ROLE_GROUP | ROLE_PUBLIC | D_ROLE |
|---------------------------|--------|---------|----------|-----------|------------|-------------|--------|
| ACCESSCTRL | * | N | * | * | N | * | * |
| BINDADD | * | N | * | * | N | * | * |
| CONNECT | * | N | * | * | N | * | * |
| CREATE_EXTERNAL_ROUTINE | * | N | * | * | N | * | * |
| CREATE_NOT_FENCED_ROUTINE | * | N | * | * | Y | * | * |
| CREATETAB | * | N | * | * | N | * | * |
| DATAACCESS | * | N | * | * | N | * | * |
| DBADM | * | N | * | * | N | * | * |
| EXPLAIN | * | N | * | * | N | * | * |
| IMPLICIT_SCHEMA | * | N | * | * | N | * | * |
| LOAD | * | N | * | * | N | * | * |
| QUIESCE_CONNECT | * | N | * | * | N | * | * |
| SECADM | * | N | * | * | N | * | * |
| SQLADM | * | N | * | * | N | * | * |
| SYSADM | * | Y | * | * | * | * | * |
| SYSCTRL | * | Y | * | * | * | * | * |
| SYSMAINT | * | Y | * | * | * | * | * |
| SYSMON | * | N | * | * | * | * | * |
| WLMADM | * | N | * | * | N | * | * |

Example 3: Retrieve all authorities special group PUBLIC has granted either directly to it or indirectly through a role

```
SELECT AUTHORITY, D_USER, D_GROUP, D_PUBLIC, ROLE_USER, ROLE_GROUP, ROLE_PUBLIC, D_ROLE
FROM TABLE (SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID ('PUBLIC', 'G')) AS T
ORDER BY AUTHORITY
```

| 1 | D_USER | D_GROUP | D_PUBLIC | ROLE_USER | ROLE_GROUP | ROLE_PUBLIC | D_ROLE |
|---------------------------|--------|---------|----------|-----------|------------|-------------|--------|
| ACCESSCTRL | * | * | N | * | * | N | * |
| BINDADD | * | * | Y | * | * | N | * |
| CONNECT | * | * | Y | * | * | N | * |
| CREATE_EXTERNAL_ROUTINE | * | * | N | * | * | N | * |
| CREATE_NOT_FENCED_ROUTINE | * | * | N | * | * | N | * |
| CREATETAB | * | * | Y | * | * | N | * |
| DATAACCESS | * | * | N | * | * | N | * |
| DBADM | * | * | N | * | * | N | * |
| EXPLAIN | * | * | N | * | * | N | * |
| IMPLICIT_SCHEMA | * | * | Y | * | * | N | * |
| LOAD | * | * | N | * | * | N | * |
| QUIESCE_CONNECT | * | * | N | * | * | N | * |
| SECADM | * | * | N | * | * | N | * |
| SQLADM | * | * | N | * | * | N | * |
| SYSADM | * | * | * | * | * | * | * |
| SYSCTRL | * | * | * | * | * | * | * |
| SYSMAINT | * | * | * | * | * | * | * |
| SYSMON | * | * | * | * | * | * | * |
| WLMADM | * | * | N | * | * | N | * |

Example 4: Retrieve all authorities role R1 has granted either directly to it or indirectly through a role. Consider in this case that role R2 was also granted to role R1.

```
SELECT AUTHORITY, D_USER, D_GROUP, D_PUBLIC, ROLE_USER, ROLE_GROUP, ROLE_PUBLIC, D_ROLE
FROM TABLE (SYSPROC.AUTH_LIST_AUTHORITIES_FOR_AUTHID ('R1', 'R')) AS T
ORDER BY AUTHORITY
```

| AUTHORITY | D_USER | D_GROUP | D_PUBLIC | ROLE_USER | ROLE_GROUP | ROLE_PUBLIC | D_ROLE |
|---------------------------|--------|---------|----------|-----------|------------|-------------|--------|
| ACCESSCTRL | * | * | * | * | * | * | N |
| BINDADD | * | * | * | * | * | * | N |
| CONNECT | * | * | * | * | * | * | N |
| CREATE_EXTERNAL_ROUTINE | * | * | * | * | * | * | N |
| CREATE_NOT_FENCED_ROUTINE | * | * | * | * | * | * | Y |

| | | | | | | | |
|-----------------|---|---|---|---|---|---|---|
| CREATETAB | * | * | * | * | * | * | N |
| DATAACCESS | * | * | * | * | * | * | N |
| DBADM | * | * | * | * | * | * | N |
| EXPLAIN | * | * | * | * | * | * | N |
| IMPLICIT_SCHEMA | * | * | * | * | * | * | N |
| LOAD | * | * | * | * | * | * | Y |
| QUIESCE_CONNECT | * | * | * | * | * | * | N |
| SECADM | * | * | * | * | * | * | N |
| SYSADM | * | * | * | * | * | * | * |
| SQLADM | * | * | * | * | * | * | N |
| SYSCTRL | * | * | * | * | * | * | * |
| SYSMAINT | * | * | * | * | * | * | * |
| SYSMON | * | * | * | * | * | * | * |
| WLMADM | * | * | * | * | * | * | N |

Usage notes

The output of AUTH_LIST_AUTHORITIES_FOR_AUTHID table function depends on the *authidtype*. For example, for an *authidtype* of USER, it returns all authorities that *authid* holds through any means:

- granted directly to the *authid*
- granted to any group (or roles granted to the group) to which *authid* belongs
- granted to any role (or roles granted to the role) granted to *authid*
- granted to PUBLIC (or roles granted to PUBLIC)

AUTH_LIST_GROUPS_FOR_AUTHID table function - Retrieve group membership list for a given authorization ID

The AUTH_LIST_GROUPS_FOR_AUTHID table function returns the list of groups of which the given authorization ID is a member.

Syntax

►►—AUTH_LIST_GROUPS_FOR_AUTHID—(—*authid*—)—►►

The schema is SYSPROC.

Table function parameter

authid

An input argument of type VARCHAR(128) that specifies the authorization ID being queried. The authorization ID can only represent a user. If *authid* does not exist, is NULL or empty string, an empty result table is returned.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve all groups that AMY belongs to.
SELECT * FROM TABLE (SYSPROC.AUTH_LIST_GROUPS_FOR_AUTHID('AMY')) AS T

The following is an example of output for this query.

GROUP
-----,.,-----
BUILD
PDXDB2

2 record(s) selected.

Usage notes

Group information returned might be different than expected for the following reasons:

- In a Windows Active Directory environment, the database manager:
 - supports one level of group nesting within a local group, except the nesting of a domain local group within a local group. For example, if *authid* belongs to the global group G1, and G1 belongs to the local group L1, the local group L1 is returned as the group for *authid*. However, if *authid* belongs to the domain local group DL1, and DL1 belongs to the local group L1, no group information is returned for *authid*.
 - does not support any nesting of global groups. For example, if *authid* belongs to the global G2, and G2 belongs to the global G3, only G2 is returned as the group for *authid*.
- The registry variable **DB2_GRP_LOOKUP** specifies which Windows security mechanism is used to enumerate the groups to which a user belongs.
- For an authorization ID that belongs to a particular domain, if the domain is not specified as part of the *authid*, and both a local and domain *authid* exist with the same name, the groups for the local authorization ID is returned.
- If the call to AUTH_LIST_GROUPS_FOR_AUTHID is for the same *authid* as the connected user, then it will return the groups for the connected user. For example, If AMY exists as a local user and as a domain user and the domain user AMY has connected to the database, then AUTH_LIST_GROUPS_FOR_AUTHID will return the groups to which the domain AMY belongs to.

Information returned

Table 186. Information returned by the AUTH_LIST_GROUPS_FOR_AUTHID table function

| Column name | Data type | Description |
|-------------|--------------|--|
| GROUP | VARCHAR(128) | The group to which the authorization ID belongs. |

AUTH_LIST_ROLES_FOR_AUTHID function - Returns the list of roles

The AUTH_LIST_ROLES_FOR_AUTHID function returns the list of roles in which the given authorization ID is a member.

Syntax

➤—AUTH_LIST_ROLES_FOR_AUTHID—(—*authid*—,—*authidtype*—)—➤

The schema is SYSPROC.

Table function parameters

authid

An input argument of type VARCHAR(128) that specifies the authorization ID being queried. The authorization ID can be a user, group or a role. If *authid* is NULL or an empty string, an empty result table is returned.

authidtype

An input argument of type VARCHAR(1) that specifies the authorization ID type being queried. If *authidtype* does not exist, is NULL or an empty string, an empty result table is returned. Possible values for *authidtype* are:

- G: Group
- R: Role
- U: User

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Information returned

Table 187. The result sets for AUTH_LIST_ROLES_FOR_AUTHID

| Column Name | Data Type | Description |
|-------------|--------------|---|
| GRANTOR | VARCHAR(128) | Grantor of the role. |
| GRANTORTYPE | CHAR(1) | Type of grantor: <ul style="list-style-type: none">• U = Grantor is an individual user |
| GRANTEE | VARCHAR(128) | User granted the role. |
| GRANTEETYPE | CHAR(1) | Type of grantee: <ul style="list-style-type: none">• G = Grantee is a group• R= Grantee is a role• U= Grantee is a user |
| ROLENAME | VARCHAR(128) | Name of the role granted to the authorization ID directly or indirectly through a group or another role. |
| CREATE_TIME | TIMESTAMP | Time when role was created. |
| ADMIN | CHAR(1) | Privilege to grant the role to, revoke the role from, or to comment on a role: <ul style="list-style-type: none">• N = Not held• Y= Held |

Example

Consider granting role INTERN to role DOCTOR and role DOCTOR to role SPECIALIST, then grant role SPECIALIST to user ALICE. ALICE belongs to group HOSPITAL and role EMPLOYEE is granted to group HOSPITAL. ALICE also belongs to special group PUBLIC and role PATIENTS is granted to PUBLIC.

Example 1: Retrieve all roles granted to user ALICE.

```
SELECT GRANTOR, GRANTORTYPE, GRANTEE, GRANTEETYPE, ROLENAME,
       CREATE_TIME, ADMIN
FROM TABLE (SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID ('ALICE', 'U') ) AS T
```

The following is an example of output for this query.

| GRANTOR | GRANTORTYPE | GRANTEE | GRANTEETYPE | ROLENAME | CREATE_TIME | ADMIN |
|---------|-------------|------------|-------------|------------|----------------------------|-------|
| ZURBIE | U | DOCTOR | R | INTERN | 2006-08-01-15.09.58.537399 | N |
| ZURBIE | U | SPECIALIST | R | DOCTOR | 2006-08-01-15.10.04.540660 | N |
| ZURBIE | U | ALICE | U | SPECIALIST | 2006-08-01-15.10.08.776218 | N |
| ZURBIE | U | HOSPITAL | G | EMPLOYEE | 2006-08-01-15.10.14.277576 | N |
| ZURBIE | U | PUBLIC | G | PATIENTS | 2006-08-01-15.10.18.878609 | N |

5 record(s) selected.

Example 2: Retrieve all roles granted to group HOSPITAL.

```
SELECT GRANTOR, GRANTORTYPE, GRANTEE, GRANTEETYPE, ROLENAME,
       CREATE_TIME, ADMIN
FROM TABLE (SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID ('HOSPITAL', 'G') ) AS T
```

The following is an example of output for this query.

| GRANTOR | GRANTORTYPE | GRANTEE | GRANTEETYPE | ROLENAME | CREATE_TIME | ADMIN |
|---------|-------------|----------|-------------|----------|----------------------------|-------|
| ZURBIE | U | HOSPITAL | G | EMPLOYEE | 2006-08-01-15.10.14.277576 | N |

1 record(s) selected.

Example 3: Retrieve all roles granted to role SPECIALIST.

```
SELECT GRANTOR, GRANTORTYPE, GRANTEE, GRANTEETYPE, ROLENAME,
       CREATE_TIME, ADMIN
FROM TABLE (SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID ('SPECIALIST', 'R') ) AS T
```

The following is an example of output for this query.

| GRANTOR | GRANTORTYPE | GRANTEE | GRANTEETYPE | ROLENAME | CREATE_TIME | ADMIN |
|---------|-------------|------------|-------------|----------|----------------------------|-------|
| ZURBIE | U | DOCTOR | R | INTERN | 2006-08-01-15.09.58.537399 | N |
| ZURBIE | U | SPECIALIST | R | DOCTOR | 2006-08-01-15.10.04.540660 | N |

2 record(s) selected.

Example 4: Retrieve all roles granted to group PUBLIC

```
SELECT GRANTOR, GRANTORTYPE, GRANTEE, GRANTEETYPE, ROLENAME,
       CREATE_TIME, ADMIN
FROM TABLE (SYSPROC.AUTH_LIST_ROLES_FOR_AUTHID ('PUBLIC', 'G') ) AS T
```

The following is an example of output for this query.

| GRANTOR | GRANTORTYPE | GRANTEE | GRANTEETYPE | ROLENAME | CREATE_TIME | ADMIN |
|---------|-------------|---------|-------------|----------|----------------------------|-------|
| ZURBIE | U | PUBLIC | G | PATIENTS | 2006-08-01-15.10.18.878609 | N |

1 record(s) selected.

Usage notes

The output of AUTH_LIST_ROLES_FOR_AUTHID table function depends on the AUTHIDTYPE:

- For a user it returns the roles granted to the user directly or indirectly through another roles, groups that the user belongs to (or PUBLIC).
- For a group it returns the roles granted to the group, directly or indirectly through another roles.
- For a role it returns the roles granted to the role, directly or indirectly through another roles.

AUTHORIZATIONIDS administrative view - Retrieve authorization IDs and types

The AUTHORIZATIONIDS administrative view returns a list of all the users, roles, and groups that exist in the database catalog of the currently connected server as a result of GRANT statements.

Each authorization ID and associated type that is returned by the view has at least one privilege, authority, or role membership. The users that are currently members of any of the groups are not included in the result.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the AUTHORIZATIONIDS administrative view
- CONTROL privilege on the AUTHORIZATIONIDS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve all authorization IDs that have been granted privileges or authorities, along with their types.

```
SELECT * FROM SYSIBMADM.AUTHORIZATIONIDS
```

The following is an example of output for this query.

| AUTHID | AUTHIDTYPE |
|----------|------------|
| PUBLIC | G |
| JESSICAE | U |
| DOCTOR | R |

3 record(s) selected.

Information returned

Table 188. Information returned by the AUTHORIZATIONIDS administrative view

| Column name | Data type | Description |
|-------------|--------------|---|
| AUTHID | VARCHAR(128) | Authorization ID that has been explicitly granted privileges or authorities. |
| AUTHIDTYPE | CHAR(1) | Authorization ID type: <ul style="list-style-type: none">• U: user• R: role• G: group |

OBJECTOWNERS administrative view - Retrieve object ownership information

The OBJECTOWNERS administrative view returns all object ownership information for every authorization ID of type USER that owns an object and that is defined in the system catalogs from the currently connected database.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the OBJECTOWNERS administrative view
- CONTROL privilege on the OBJECTOWNERS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve all object ownership information for object schema 'THERESAX'.

```
SELECT SUBSTR(OWNER,1,10) AS OWNER, OWNERTYPE,
       SUBSTR(OBJECTNAME,1,30) AS OBJECTNAME,
       SUBSTR(OBJECTSCHEMA,1,10) AS OBJECTSCHEMA, OBJECTTYPE
FROM SYSIBMADM.OBJECTOWNERS WHERE OBJECTSCHEMA='THERESAX'
```

The following is an example of output for this query.

| OWNER | OWNERTYPE | OBJECTNAME | OBJECTSCHEMA | OBJECTTYPE |
|----------|-----------|-------------------|--------------|------------------|
| ----- | ----- | ----- | ----- | ----- |
| THERESAX | U | MIN_SALARY | THERESAX | TRIGGER |
| THERESAX | U | POLICY_IR | SYSTOOLS | TRIGGER |
| THERESAX | U | CUSTOMER | THERESAX | XML SCHEMA |
| THERESAX | U | DB2DETAILDEADLOCK | | EVENTMONITORS |
| THERESAX | U | SAMPSEQUENCE | THERESAX | SEQUENCE |
| THERESAX | U | SQL0F00 | NULLID | PACKAGE |
| ... | | | | |
| THERESAX | U | HI_OBJ_UNIQ | SYSTOOLS | TABLE CONSTRAINT |

257 record(s) selected.

Information returned

Table 189. Information returned by the OBJECTOWNERS administrative view

| Column name | Data type | Description |
|--------------|--------------|--|
| OWNER | VARCHAR(128) | Authorization ID that owns this object. |
| OWNERTYPE | VARCHAR(1) | Authorization ID type: <ul style="list-style-type: none">• U: user• S: system |
| OBJECTNAME | VARCHAR(128) | object_name - Object name monitor element |
| OBJECTSCHEMA | VARCHAR(128) | object_schema - Object schema monitor element |
| OBJECTTYPE | VARCHAR(24) | Database object type. |

PRIVILEGES administrative view - Retrieve privilege information

The PRIVILEGES administrative view returns all explicit privileges for all authorization IDs defined in the system catalogs from the currently connected database.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the PRIVILEGES administrative view
- CONTROL privilege on the PRIVILEGES administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve the privilege granted along with the object name, schema and type, for all authorization IDs.

```
SELECT AUTHID, PRIVILEGE, OBJECTNAME, OBJECTSCHEMA, OBJECTTYPE
FROM SYSIBMADM.PRIVILEGES
```

The following is an example of output for this query.

| AUTHID | PRIVILEGE | OBJECTNAME | OBJECTSCHEMA | OBJECTTYPE |
|----------|-----------|--------------------|--------------|------------|
| ----- | ----- | ----- | ----- | ----- |
| JESSICAE | EXECUTE | SQL0F00 | NULLID | PACKAGE |
| PUBLIC | EXECUTE | SYSSH201 | NULLID | PACKAGE |
| JESSICAE | EXECUTE | SYSSH202 | NULLID | PACKAGE |
| PUBLIC | EXECUTE | SYSSH202 | NULLID | PACKAGE |
| DOCTOR | EXECUTE | PKG0123 | NULLID | PACKAGE |
| ... | | | | |
| PUBLIC | EXECUTE | SQL051109185227800 | SYSPROC | FUNCTION |
| JESSICAE | EXECUTE | SQL051109185227801 | SYSPROC | FUNCTION |

| | | | | |
|----------|----------|--------------------|----------|-----------|
| PUBLIC | EXECUTE | SQL051109185227801 | SYSPROC | FUNCTION |
| JESSICAE | EXECUTE | SQL051109185227838 | SYSPROC | FUNCTION |
| PUBLIC | EXECUTE | SQL051109185227838 | SYSPROC | FUNCTION |
| ... | | | | |
| PUBLIC | EXECUTE | LIST_SRVR_TYPES | SYSPROC | PROCEDURE |
| PUBLIC | EXECUTE | LIST_SRVR_VERSIONS | SYSPROC | PROCEDURE |
| PUBLIC | EXECUTE | LIST_WRAP_OPTIONS | SYSPROC | PROCEDURE |
| PUBLIC | EXECUTE | LIST_SRVR_OPTIONS | SYSPROC | PROCEDURE |
| ... | | | | |
| SYSTEM | | POLICY_UNQ | SYSTOOLS | INDEX |
| PUBLIC | CREATEIN | | NULLID | SCHEMA |
| PUBLIC | UPDATE | COLUMNS | SYSSTAT | VIEW |
| PUBLIC | UPDATE | COLGROUPS | SYSSTAT | VIEW |
| ... | | | | |

Information returned

Table 190. Information returned by the PRIVILEGES administrative view

| Column name | Data type | Description |
|--------------|--------------|---|
| AUTHID | VARCHAR(128) | Authorization ID that has been explicitly granted this privilege. |
| AUTHIDTYPE | CHAR(1) | Authorization ID type: <ul style="list-style-type: none"> • U: user • R: role • G: group |
| PRIVILEGE | VARCHAR(11) | Privilege that has been explicitly granted to this authorization ID. |
| GRANTABLE | VARCHAR(1) | Indicates if the privilege is grantable: <ul style="list-style-type: none"> • Y: Grantable • N: Not grantable |
| OBJECTNAME | VARCHAR(128) | object_name - Object name monitor element |
| OBJECTSCHEMA | VARCHAR(128) | object_schema - Object schema monitor element |
| OBJECTTYPE | VARCHAR(24) | Database object type. |

Snapshot routines and views

This grouping of routines and views can be used to retrieve information about the database and any connected applications at a specific time.

QUERY_PREP_COST administrative view - Retrieve statement prepare time information

The QUERY_PREP_COST administrative view returns a list of statements with information about the time required to prepare the statement.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the QUERY_PREP_COST administrative view

- CONTROL privilege on the QUERY_PREP_COST administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a report on the queries with the highest percentage of time spent on preparing.

```
SELECT NUM_EXECUTIONS, AVERAGE_EXECUTION_TIME_S, PREP_TIME_PERCENT,
       SUBSTR(STMT_TEXT, 1, 30) AS STMT_TEXT, DBPARTITIONNUM
FROM SYSIBMADM.QUERY_PREP_COST ORDER BY PREP_TIME_PERCENT
```

The following is an example of output for this query.

```
NUM_EXECUTIONS      AVERAGE_EXECUTION_TIME_S ...
-----
1                      25 ...
```

1 record(s) selected.

Output for this query (continued).

```
... PREP_TIME_PERCENT STMT_TEXT                      DBPARTITIONNUM
... -----
...                0.0 select * from dbuser.employee                0
```

Usage notes

When selecting from the view, an order by clause can be used to identify queries with the highest prep cost. You can examine this view to see how frequently a query is run as well as the average execution time for each of these queries. If the time it takes to compile and optimize a query is almost as long as it takes for the query to execute, you might want to look at the optimization class that you are using. Lowering the optimization class might make the query complete optimization more rapidly and therefore return a result sooner. However, if a query takes a significant amount of time to prepare yet is executed thousands of times (without being prepared again) then the optimization class might not be an issue.

Information returned

Table 191. Information returned by the QUERY_PREP_COST administrative view

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-----------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time the report was generated. |
| NUM_EXECUTIONS | BIGINT | num_executions - Statement executions |
| AVERAGE_EXECUTION_TIME_S | BIGINT | Average execution time (in seconds). |
| AVERAGE_EXECUTION_TIME_MS | BIGINT | Average execution time (fractional, in microseconds). |

Table 191. Information returned by the `QUERY_PREP_COST` administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------|--------------|--|
| PREP_TIME_MS | BIGINT | prep_time_worst - Statement worst preparation time (in milliseconds). |
| PREP_TIME_PERCENT | DECIMAL(8,2) | Percent of execution time spent on preparation. Calculated as prep time divided by total execution time. |
| STMT_TEXT | CLOB(2 M) | stmt_text - SQL statement text |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL table function - Retrieve dynsql logical group snapshot information

The `SNAPDYN_SQL` administrative view and the `SNAP_GET_DYN_SQL` table function return snapshot information from the dynsql logical data group.

The “`MON_GET_PKG_CACHE_STMT` table function - Get package cache statement metrics” on page 686 returns a point-in-time view of both static and dynamic SQL statements in the database package cache. If you use this table function, you can take advantage of the monitoring infrastructure introduced in Version 9.7.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “`SNAPDYN_SQL` administrative view”
- “`SNAP_GET_DYN_SQL` table function” on page 942

SNAPDYN_SQL administrative view

This administrative view allows you to retrieve dynsql logical group snapshot information for the currently connected database.

This view returns information equivalent to the **GET SNAPSHOT FOR DYNAMIC SQL ON database-alias** CLP command.

The schema is `SYSIBMADM`.

Refer to Table 192 on page 944 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- `SELECT` privilege on the `SNAPDYN_SQL` administrative view
- `CONTROL` privilege on the `SNAPDYN_SQL` administrative view
- `DATAACCESS` authority
- `DBADM` authority

- SQLADM authority

One of the following authorizations is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DYN_SQL table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of dynamic SQL run on the currently connected database, ordered by the number of rows read.

```
SELECT PREP_TIME_WORST, NUM_COMPILATIONS, SUBSTR(STMT_TEXT, 1, 60)
  AS STMT_TEXT, DBPARTITIONNUM
  FROM SYSIBMADM.SNAPDYN_SQL ORDER BY ROWS_READ
```

The following is an example of output from this query.

| PREP_TIME_WORST | NUM_COMPILATIONS | ... |
|-----------------|------------------|-----|
| ----- | ----- | ... |
| 98 | 1 | ... |
| 9 | 1 | ... |
| 0 | 0 | ... |
| 0 | 1 | ... |
| 0 | 1 | ... |
| 0 | 1 | ... |
| 0 | 1 | ... |
| 0 | 1 | ... |
| 40 | 1 | ... |

9 record(s) selected.

Output from this query (continued).

```
... STMT_TEXT
... -----
... select prep_time_worst, num_compilations, substr(stmt_text, ...
... select * from dbuser.employee ...
... SET CURRENT LOCALE LC_CTYPE = 'en_US' ...
... select prep_time_worst, num_compilations, substr(stmt_text, ...
... select prep_time_worst, num_compilations, substr(stmt_text, ...
... select * from dbuser.employee ...
... insert into dbuser.employee values(1) ...
... select * from dbuser.employee ...
... insert into dbuser.employee values(1) ...
```

Output from this query (continued).

```

... DBPARTITIONNUM
... -----
...          0
...          0
...          0
...          2
...          1
...          2
...          2
...          1
...          0

```

SNAP_GET_DYN_SQL table function

The SNAP_GET_DYN_SQL table function returns the same information as the SNAPDYN_SQL administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

This table function returns information equivalent to the **GET SNAPSHOT FOR DYNAMIC SQL ON database-alias** CLP command.

Refer to Table 192 on page 944 for a complete list of information that can be returned.

Syntax

```

-- SNAP_GET_DYN_SQL ( --dbname-- [ , member ] )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current member, or -2 for an aggregate of all active members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from members where the database is active.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DYN_SQL table function takes a snapshot for the currently connected database and database member number.

Important: The SYSPROC.SNAP_WRITE_FILE procedure is deprecated in Version 10.5 and might be removed in a future release. For more information, see "SNAP_WRITE_FILE procedure" in *Administrative Routines and Views*.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DYN_SQL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of dynamic SQL run on the currently connected database, ordered by the number of rows read.

```
SELECT PREP_TIME_WORST, NUM_COMPILATIONS, SUBSTR(STMT_TEXT, 1, 60)
      AS STMT_TEXT FROM TABLE(SNAP_GET_DYN_SQL('','-1')) as T
      ORDER BY ROWS_READ
```

The following is an example of output from this query.

| PREP_TIME_WORST | NUM_COMPILATIONS | ... |
|-----------------|------------------|-----|
| 0 | 0 | ... |
| 49 | 1 | ... |
| 0 | 0 | ... |
| 46 | 1 | ... |
| 0 | 0 | ... |
| 0 | 0 | ... |
| 0 | 0 | ... |
| 29 | 1 | ... |
| 0 | 0 | ... |
| 0 | 0 | ... |
| 10 | 1 | ... |
| 0 | 0 | ... |
| 4 | 0 | ... |
| 53 | 0 | ... |
| 0 | 0 | ... |
| 6 | 1 | ... |
| 334 | 0 | ... |
| 0 | 0 | ... |
| 5 | 0 | ... |
| 10 | 0 | ... |
| 599 | 0 | ... |
| 15 | 1 | ... |
| 7 | 0 | ... |

23 record(s) selected.

Output from this query (continued).

```
... STMT_TEXT
...
... SET :HV00017 :HI00017 = RPAD(VARCHAR(:HV00035 :HI00035 ),
```

```

... SELECT COLNAME, TYPENAME FROM SYSCAT.COLUMNS WHERE TABNAME=
... DECLARE RES CURSOR WITH RETURN TO CALLER FOR SELECT R.TEXT F
... SELECT PREP_TIME_WORST, NUM_COMPILATIONS, SUBSTR(STMT_TEXT,
... VALUES (:HV00026 :HI00026 + 1, :HV00024 :HI00024 + 1) IN
... VALUES (:HV00035 :HI00035 + 1, :HV00024 :HI00024 + 1) IN
... VALUES (1) INTO :HV00035 :HI00035
... SELECT TRIGNAME FROM SYSCAT.TRIGGERS WHERE TABNAME='POLICY'
... VALUES (:HV00024 :HI00024 +1, :HV00022 :HI00022 +1) INTO :
... VALUES (1, CARDINALITY(CAST(:HV00040 :HI00040 AS "SYSIBMAD
... CALL SYSPROC.SYSINSTALLOBJECTS('POLICY','V','I','I')
... SET :HV00017 :HI00017 = RPAD(VARCHAR(:HV00035 :HI00035 ),
... drop event monitor act
... SELECT TABSCHEMA, TABNAME, TYPE, STATUS, TBSPACEID, PROPERTY
... CALL SAVE_EXEC_INFO (CAST(:HV00040 :HI00040 AS "SYSIBMADM"
... SET CURRENT LOCK TIMEOUT 5
... SELECT TABNAME FROM SYSCAT.PERIODS WHERE PERIODNAME = 'SYSTE
... SELECT ARRAY_AGG(P.EXECUTABLE_ID ORDER BY M.IO_WAIT_TIME DES
... SET CURRENT ISOLATION RESET
... CALL monreport.pkgcache()
... SELECT A.SPECIFICNAME FROM SYSCAT.ROUTINES A WHERE (A.FENCED
... SELECT POLICY FROM SYSTOOLS.POLICY WHERE MED='DB2CommonMED'
... VALUES 0

```

23 record(s) selected.

After running a workload, user can use the following query with the table function.

```

SELECT STATS_FABRICATE_TIME,SYNC_RUNSTATS_TIME
FROM TABLE (SNAP_GET_DYN_SQL(TmytestdbT, -1))
AS SNAPDB

```

| STATS_FABRICATE_TIME | SYNC_RUNSTATS_TIME |
|----------------------|--------------------|
| 2 | 12 |
| 1 | 30 |

For the view based on this table function:

```

SELECT STATS_FABRICATE_TIME,SYNC_RUNSTATS_TIME
FROM SYSIBMADM.SNAPDYN_SQL

```

| STATS_FABRICATE_TIME | SYNC_RUNSTATS_TIME |
|----------------------|--------------------|
| 5 | 10 |
| 3 | 20 |

2 record(s) selected.

Information returned

Table 192. Information returned by the SNAPDYN_SQL administrative view and the SNAP_GET_DYN_SQL table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| NUM_EXECUTIONS | BIGINT | num_executions - Statement executions |
| NUM_COMPILATIONS | BIGINT | num_compilations - Statement compilations |
| PREP_TIME_WORST | BIGINT | prep_time_worst - Statement worst preparation time |
| PREP_TIME_BEST | BIGINT | prep_time_best - Statement best preparation time |

Table 192. Information returned by the SNAPDYN_SQL administrative view and the SNAP_GET_DYN_SQL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal rows deleted |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal rows inserted |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal rows updated |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| STMT_SORTS | BIGINT | stmt_sorts - Statement sorts |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total sort time |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| TOTAL_EXEC_TIME | BIGINT | total_exec_time - Elapsed statement execution time (in seconds)* |
| TOTAL_EXEC_TIME_MS | BIGINT | total_exec_time - Elapsed statement execution time (fractional, in microseconds)* |
| TOTAL_USR_CPU_TIME | BIGINT | total_usr_cpu_time - Total user CPU for a statement (in seconds)* |
| TOTAL_USR_CPU_TIME_MS | BIGINT | total_usr_cpu_time - Total user CPU for a statement (fractional, in microseconds)* |
| TOTAL_SYS_CPU_TIME | BIGINT | total_sys_cpu_time - Total system CPU for a statement (in seconds)* |

Table 192. Information returned by the SNAPDYN_SQL administrative view and the SNAP_GET_DYN_SQL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--|-----------|--|
| TOTAL_SYS_CPU_TIME_MS | BIGINT | total_sys_cpu_time - Total system CPU for a statement (fractional, in microseconds)* |
| STMT_TEXT | CLOB(2 M) | stmt_text - SQL statement text |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| STATS_FABRICATE_TIME | BIGINT | The total time (in milliseconds) spent by system to create needed statistics without table or index scan during query compilation for a dynamic statement. |
| SYNC_RUNSTATS_TIME | BIGINT | The total time (in milliseconds) spent on synchronous statistics-collect activities during query compilation for a dynamic statement. |
| MEMBER | SMALLINT | member - Database member monitor element |
| <p>* To calculate the total time spent for the monitor element that this column is based on, you must add the full seconds reported in the column for this monitor element that ends with _S to the fractional seconds reported in the column for this monitor element that ends with _MS, using the following formula: $(\text{monitor-element-name_S} \times 1,000,000 + \text{monitor-element-name_MS}) \div 1,000,000$. For example, $(\text{ELAPSED_EXEC_TIME_S} \times 1,000,000 + \text{ELAPSED_EXEC_TIME_MS}) \div 1,000,000$.</p> | | |

SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function - Retrieve subsection logical monitor group snapshot information

The SNAPSUBSECTION administrative view and the SNAP_GET_SUBSECTION table function return information about application subsections, namely the subsection logical monitor grouping.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPSUBSECTION administrative view”
- “SNAP_GET_SUBSECTION table function” on page 947

SNAPSUBSECTION administrative view

This administrative view allows you to retrieve subsection logical monitor group snapshot information for the currently connected database.

Used with the MON_CURRENT_SQL administrative view and the MON_GET_AGENT, MON_GET_CONNECTION, MON_GET_MEMORY_POOL, and MON_GET_MEMORY_SET table functions, the SNAPSUBSECTION administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS on database-alias** CLP command, but retrieves data from all database members.

The schema is SYSIBMADM.

Refer to Table 193 on page 949 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPSUBSECTION administrative view
- CONTROL privilege on the SNAPSUBSECTION administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_SUBSECTION table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Get status for subsections executing on all database members.

```
SELECT DB_NAME, STMT_TEXT, SS_STATUS, DBPARTITIONNUM
FROM SYSIBMADM.SNAPSUBSECTION
ORDER BY DB_NAME, SS_STATUS, DBPARTITIONNUM
```

The following is an example of output from this query.

| DB_NAME | STMT_TEXT | SS_STATUS | DBPARTITIONNUM |
|---------|------------------------|-----------|----------------|
| ... | ----- | ... | ----- |
| SAMPLE | select * from EMPLOYEE | EXEC | 0 |
| SAMPLE | select * from EMPLOYEE | EXEC | 1 |

SNAP_GET_SUBSECTION table function

The SNAP_GET_SUBSECTION table function returns the same information as the SNAPSUBSECTION administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Refer to Table 193 on page 949 for a complete list of information that can be returned.

Used with the SNAP_GET_AGENT, SNAP_GET_AGENT_MEMORY_POOL, SNAP_GET_APPL, SNAP_GET_APPL_INFO and SNAP_GET_STMT table functions, the SNAP_GET_SUBSECTION table function provides information

equivalent to the **GET SNAPSHOT FOR ALL APPLICATIONS** CLP command, but retrieves data from all database members.

Syntax

►► SNAP_GET_SUBSECTION ((*dbname* [, *member*])) ►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_SUBSECTION table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_SUBSECTION table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Get status for subsections executing on all database members.

```
SELECT DB_NAME, STMT_TEXT, SS_STATUS, DBPARTITIONNUM
FROM TABLE(SYSPROC.SNAP_GET_SUBSECTION( '', 0 )) as T
ORDER BY DB_NAME, SS_STATUS, DBPARTITIONNUM
```

The following is an example of output from this query.

| DB_NAME | STMT_TEXT | SS_STATUS | DBPARTITIONNUM |
|---------|------------------------|-----------|----------------|
| SAMPLE | select * from EMPLOYEE | EXEC | 0 |
| SAMPLE | select * from EMPLOYEE | EXEC | 1 |

Information returned

Table 193. Information returned by the SNAPSUBSECTION administrative view and the SNAP_GET_SUBSECTION table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| STMT_TEXT | CLOB(16 M) | stmt_text - SQL statement text |
| SS_EXEC_TIME | BIGINT | ss_exec_time - Subsection execution elapsed time |
| TQ_TOT_SEND_SPILLS | BIGINT | tq_tot_send_spills - Total number of table queue buffers overflowed |
| TQ_CUR_SEND_SPILLS | BIGINT | tq_cur_send_spills - Current number of table queue buffers overflowed |
| TQ_MAX_SEND_SPILLS | BIGINT | tq_max_send_spills - Maximum number of table queue buffers overflows |
| TQ_ROWS_READ | BIGINT | tq_rows_read - Number of rows read from table queues |
| TQ_ROWS_WRITTEN | BIGINT | tq_rows_written - Number of rows written to table queues |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| SS_USR_CPU_TIME_S | BIGINT | ss_usr_cpu_time - User CPU time used by subsection (in seconds)* |
| SS_USR_CPU_TIME_MS | BIGINT | ss_usr_cpu_time - User CPU time used by subsection (fractional, in microseconds)* |
| SS_SYS_CPU_TIME_S | BIGINT | ss_sys_cpu_time - System CPU time used by subsection (in seconds)* |
| SS_SYS_CPU_TIME_MS | BIGINT | ss_sys_cpu_time - System CPU time used by subsection (fractional, in microseconds)* |
| SS_NUMBER | INTEGER | ss_number - Subsection number |

Table 193. Information returned by the SNAPSUBSECTION administrative view and the SNAP_GET_SUBSECTION table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--|-------------|--|
| SS_STATUS | VARCHAR(20) | ss_status - Subsection status. This interface returns a text identifier based on defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • EXEC • TQ_WAIT_TO_RCV • TQ_WAIT_TO_SEND • COMPLETED |
| SS_NODE_NUMBER | SMALLINT | ss_node_number - Subsection node number |
| TQ_NODE_WAITED_FOR | SMALLINT | tq_node_waited_for - Waited for node on a table queue |
| TQ_WAIT_FOR_ANY | INTEGER | tq_wait_for_any - Waiting for any node to send on a table queue |
| TQ_ID_WAITING_ON | INTEGER | tq_id_waiting_on - Waited on node on a table queue |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| <p>* To calculate the total time spent for the monitor element that this column is based on, you must add the full seconds reported in the column for this monitor element that ends with _S to the fractional seconds reported in the column for this monitor element that ends with _MS, using the following formula: $(\text{monitor-element-name_S} \times 1,000,000 + \text{monitor-element-name_MS}) \div 1,000,000$. For example, $(\text{ELAPSED_EXEC_TIME_S} \times 1,000,000 + \text{ELAPSED_EXEC_TIME_MS}) \div 1,000,000$.</p> | | |

SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function - Retrieve database snapshot switch state information

The SNAPSWITCHES administrative view and the SNAP_GET_SWITCHES table function return information about the database snapshot switch state.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPSWITCHES administrative view”
- “SNAP_GET_SWITCHES table function” on page 952

SNAPSWITCHES administrative view

This view provides the data equivalent to the **GET DBM MONITOR SWITCHES CLP** command.

The schema is SYSIBMADM.

Refer to Table 194 on page 953 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPSWITCHES administrative view
- CONTROL privilege on the SNAPSWITCHES administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_SWITCHES table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve DBM monitor switches state information for all database members.

```
SELECT UOW_SW_STATE, STATEMENT_SW_STATE, TABLE_SW_STATE, BUFFPOOL_SW_STATE,
       LOCK_SW_STATE, SORT_SW_STATE, TIMESTAMP_SW_STATE,
       DBPARTITIONNUM FROM SYSIBMADM.SNAPSWITCHES
```

The following is an example of output from this query.

| UOW_SW_STATE | STATEMENT_SW_STATE | TABLE_SW_STATE | BUFFPOOL_SW_STATE | ... |
|--------------|--------------------|----------------|-------------------|-----|
| 0 | 0 | 0 | 0 | ... |
| 0 | 0 | 0 | 0 | ... |
| 0 | 0 | 0 | 0 | ... |
| ... | | | | |

3 record selected.

Output from this query (continued).

| ... | LOCK_SW_STATE | SORT_SW_STATE | TIMESTAMP_SW_STATE | DBPARTITIONNUM |
|-----|---------------|---------------|--------------------|----------------|
| ... | 1 | 0 | 1 | 0 |
| ... | 1 | 0 | 1 | 1 |
| ... | 1 | 0 | 1 | 2 |

SNAP_GET_SWITCHES table function

The SNAP_GET_SWITCHES table function returns the same information as the SNAPSWITCHES administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

This table function provides the data equivalent to the **GET DBM MONITOR SWITCHES** CLP command.

Refer to Table 194 on page 953 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_SWITCHES (*member*) ►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_SWITCHES table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_SWITCHES table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Retrieve DBM monitor switches state information for the current database member.

```
SELECT UOW_SW_STATE, STATEMENT_SW_STATE, TABLE_SW_STATE,
       BUFFPOOL_SW_STATE, LOCK_SW_STATE, SORT_SW_STATE, TIMESTAMP_SW_STATE
FROM TABLE(SNAP_GET_SWITCHES(-1)) AS T
```

The following is an example of output from this query.

```
UOW_SW_STATE STATEMENT_SW_STATE TABLE_SW_STATE...
-----
1 1 1...
...
1 record(s) selected. ...
```

Output from this query (continued).

```
... BUFFPOOL_SW_STATE LOCK_SW_STATE SORT_SW_STATE TIMESTAMP_SW_STATE
... -----
... 1 1 0 1
```

Information returned

Table 194. Information returned by the SNAPSWITCHES administrative view and the SNAP_GET_SWITCHES table function

| Column name | Data type | Description |
|--------------------|-----------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| UOW_SW_STATE | SMALLINT | State of the unit of work monitor recording switch (0 or 1). |
| UOW_SW_TIME | TIMESTAMP | If the unit of work monitor recording switch is on, the date and time that this switch was turned on. |
| STATEMENT_SW_STATE | SMALLINT | State of the SQL statement monitor recording switch (0 or 1). |
| STATEMENT_SW_TIME | TIMESTAMP | If the SQL statement monitor recording switch is on, the date and time that this switch was turned on. |
| TABLE_SW_STATE | SMALLINT | State of the table activity monitor recording switch (0 or 1). |
| TABLE_SW_TIME | TIMESTAMP | If the table activity monitor recording switch is on, the date and time that this switch was turned on. |
| BUFFPOOL_SW_STATE | SMALLINT | State of the buffer pool activity monitor recording switch (0 or 1). |
| BUFFPOOL_SW_TIME | TIMESTAMP | If the buffer pool activity monitor recording switch is on, the date and time that this switch was turned on. |
| LOCK_SW_STATE | SMALLINT | State of the lock monitor recording switch (0 or 1). |

Table 194. Information returned by the SNAPSWITCHES administrative view and the SNAP_GET_SWITCHES table function (continued)

| Column name | Data type | Description |
|--------------------|-----------|--|
| LOCK_SW_TIME | TIMESTAMP | If the lock monitor recording switch is on, the date and time that this switch was turned on. |
| SORT_SW_STATE | SMALLINT | State of the sorting monitor recording switch (0 or 1). |
| SORT_SW_TIME | TIMESTAMP | If the sorting monitor recording switch is on, the date and time that this switch was turned on. |
| TIMESTAMP_SW_STATE | SMALLINT | State of the timestamp monitor recording switch (0 or 1) |
| TIMESTAMP_SW_TIME | TIMESTAMP | If the timestamp monitor recording switch is on, the date and time that this switch was turned on. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPTAB_REORG administrative view and SNAP_GET_TAB_REORG table function - Retrieve table reorganization snapshot information

The SNAPTAB_REORG administrative view and the SNAP_GET_TAB_REORG table function return table reorganization information.

If no tables have been reorganized, 0 rows are returned. When a data partitioned table is reorganized, one record for each data partition is returned. If only a specific data partition of a data partitioned table is reorganized, only a record for the partition is returned.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPTAB_REORG administrative view”
- “SNAP_GET_TAB_REORG table function” on page 956

SNAPTAB_REORG administrative view

This administrative view allows you to retrieve table reorganization snapshot information for the currently connected database.

Used with the SNAPTAB administrative view, the SNAPTAB_REORG administrative view provides the data equivalent to the **GET SNAPSHOT FOR TABLES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 195 on page 958 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPTAB_REORG administrative view
- CONTROL privilege on the SNAPTAB_REORG administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_TAB_REORG table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Select details on reorganization operations for all database members on the currently connected database.

```
SELECT SUBSTR(TABNAME, 1, 15) AS TAB_NAME, SUBSTR(TABSCHEMA, 1, 15)
      AS TAB_SCHEMA, REORG_PHASE, SUBSTR(REORG_TYPE, 1, 20) AS REORG_TYPE,
      REORG_STATUS, REORG_COMPLETION, DBPARTITIONNUM
FROM SYSIBMADM.SNAPTAB_REORG ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| TAB_NAME | TAB_SCHEMA | REORG_PHASE | ... |
|----------|------------|-------------|-----|
| EMPLOYEE | DBUSER | REPLACE | ... |
| EMPLOYEE | DBUSER | REPLACE | ... |
| EMPLOYEE | DBUSER | REPLACE | ... |

3 record(s) selected.

Output from this query (continued).

| ... REORG_TYPE | REORG_STATUS | REORG_COMPLETION | DBPARTITIONNUM |
|--------------------------|--------------|------------------|----------------|
| ... RECLAIM+OFFLINE+ALLO | COMPLETED | SUCCESS | 0 |
| ... RECLAIM+OFFLINE+ALLO | COMPLETED | SUCCESS | 1 |
| ... RECLAIM+OFFLINE+ALLO | COMPLETED | SUCCESS | 2 |

Select all information about a reorganization operation to reclaim extents from a multidimensional clustering (MDC) or insert time clustering (ITC) table from the SNAPTAB_REORG administrative view.

```
db2 -v "select * from sysibmadm.snaptab_reorg"
```

| TABNAME | REORG_PHASE | REORG_MAX_PHASE | REORG_TYPE |
|--------------|------------------|----------------------------|-----------------------------|
| T1 | RELEASE | 3 | RECLAIM_EXTENTS+ALLOW_WRITE |
| REORG_STATUS | REORG_COMPLETION | REORG_START | REORG_END |
| COMPLETED | SUCCESS | 2008-09-24-14.35.30.734741 | 2008-09-24-14.35.31.460674 |

SNAP_GET_TAB_REORG table function

The SNAP_GET_TAB_REORG table function returns the same information as the SNAPTAB_REORG administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_TAB table function, the SNAP_GET_TAB_REORG table function provides the data equivalent to the **GET SNAPSHOT FOR TABLES ON database-alias** CLP command.

Refer to Table 195 on page 958 for a complete list of information that can be returned.

Syntax

```

--SNAP_GET_TAB_REORG--(---dbname---[ , member ]---)

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TAB_REORG table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_TAB_REORG table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Select details on reorganization operations for database member 1 on the currently connected database.

```
SELECT SUBSTR(TABNAME, 1, 15) AS TAB_NAME, SUBSTR(TABSCHEMA, 1, 15)
      AS TAB_SCHEMA, REORG_PHASE, SUBSTR(REORG_TYPE, 1, 20) AS REORG_TYPE,
      REORG_STATUS, REORG_COMPLETION, DBPARTITIONNUM
FROM TABLE( SNAP_GET_TAB_REORG('', 1)) AS T
```

The following is an example of output from this query.

| TAB_NAME | TAB_SCHEMA | REORG_PHASE | REORG_TYPE | ... |
|-----------------------|------------|-------------|----------------------|-----|
| EMPLOYEE | DBUSER | REPLACE | RECLAIM+OFFLINE+ALLO | ... |
| 1 record(s) selected. | | | | ... |

Output from this query (continued).

| ... | REORG_STATUS | REORG_COMPLETION | DBPARTITIONNUM |
|-----|--------------|------------------|----------------|
| ... | COMPLETED | SUCCESS | 1 |
| ... | | | |

Select all information about a reorganization operation to reclaim extents from a multidimensional clustering (MDC) or insert time clustering (ITC) table using the SNAP_GET_TAB_REORG table function.

```
db2 -v "select * from table(snap_get_tab_reorg(''))"
```

| TABNAME | REORG_PHASE | REORG_MAX_PHASE | REORG_TYPE |
|---------|-------------|-----------------|-----------------------------|
| T1 | RELEASE | 3 | RECLAIM_EXTENTS+ALLOW_WRITE |

| REORG_STATUS | REORG_COMPLETION | REORG_START | REORG_END |
|--------------|------------------|----------------------------|----------------------------|
| COMPLETED | SUCCESS | 2008-09-24-14.35.30.734741 | 2008-09-24-14.35.31.460674 |

Information returned

Table 195. Information returned by the `SNAPTAB_REORG` administrative view and the `SNAP_GET_TAB_REORG` table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABNAME | VARCHAR (128) | table_name - Table name |
| TABSCHEMA | VARCHAR (128) | table_schema - Table schema name |
| PAGE_REORGS | BIGINT | page_reorgs - Page reorganizations |
| REORG_PHASE | VARCHAR (16) | reorg_phase - Table reorganize phase. This interface returns a text identifier based on defines in <code>sqlmon.h</code> and is one of: <ul style="list-style-type: none">• BUILD• DICT_SAMPLE• INDEX_RECREATE• REPLACE• SORT• SCAN• DRAIN• RELEASE or SORT+DICT_SAMPLE. |
| REORG_MAX_PHASE | INTEGER | reorg_max_phase - Maximum table reorganize phase |
| REORG_CURRENT_COUNTER | BIGINT | reorg_current_counter - Table reorganize progress |
| REORG_MAX_COUNTER | BIGINT | reorg_max_counter - Total amount of table reorganization |

Table 195. Information returned by the SNAPTAB_REORG administrative view and the SNAP_GET_TAB_REORG table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------|------------------|--|
| REORG_TYPE | VARCHAR
(128) | <p>reorg_type - Table reorganize attributes. This interface returns a text identifier using a combination of the following identifiers separated by '+':</p> <p>Either:</p> <ul style="list-style-type: none"> • RECLAIM • RECLUSTER • RECLAIM_EXTS <p>and either:</p> <ul style="list-style-type: none"> • +OFFLINE • +ONLINE <p>If access mode is specified, it is one of:</p> <ul style="list-style-type: none"> • +ALLOW_NONE • +ALLOW_READ • +ALLOW_WRITE <p>If offline and RECLUSTER option, one of:</p> <ul style="list-style-type: none"> • +INDEXSCAN • +TABLESCAN <p>If offline, one of:</p> <ul style="list-style-type: none"> • +LONGLOB • +DATAONLY <p>If offline, and option is specified, any of:</p> <ul style="list-style-type: none"> • +CHOOSE_TEMP • +KEEPPDICTIONARY • +RESETDICTIONARY <p>If online, and option is specified:</p> <ul style="list-style-type: none"> • +NOTRUNCATE <p>Example 1: If a REORG TABLE TEST.EMPLOYEE was run, the following would be displayed:</p> <p>RECLAIM+OFFLINE+ALLOW_READ+DATAONLY+KEEPPDICTIONARY</p> <p>Example 2: If a REORG TABLE TEST.EMPLOYEE INDEX EMPIDX INDEXSCAN was run, then the following would be displayed:</p> <p>RECLUSTER+OFFLINE+ALLOW_READ+INDEXSCAN+DATAONLY+KEEPPDICTIONARY</p> |

Table 195. Information returned by the `SNAPTAB_REORG` administrative view and the `SNAP_GET_TAB_REORG` table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|-----------------|---|
| REORG_STATUS | VARCHAR
(10) | reorg_status - Table reorganize status. This interface returns a text identifier based on defines in <code>sqlmon.h</code> and is one of: <ul style="list-style-type: none"> • COMPLETED • PAUSED • STARTED • STOPPED • TRUNCATE |
| REORG_COMPLETION | VARCHAR
(10) | reorg_completion - Table reorganization completion flag. This interface returns a text identifier, based on defines in <code>sqlmon.h</code> and is one of: <ul style="list-style-type: none"> • FAIL • SUCCESS |
| REORG_START | TIMESTAMP | reorg_start - Table reorganize start time |
| REORG_END | TIMESTAMP | reorg_end - Table reorganize end time |
| REORG_PHASE_START | TIMESTAMP | reorg_phase_start - Table reorganize phase start time |
| REORG_INDEX_ID | BIGINT | reorg_index_id - Index used to reorganize the table |
| REORG_TBSPC_ID | BIGINT | reorg_tbsp_id - Table space where table is reorganized |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition identifier. For a non-partitioned table, this element will be NULL. |
| REORG_ROWSCOMPRESSED | BIGINT | reorg_rows_compressed - Rows compressed |
| REORG_ROWSREJECTED | BIGINT | reorg_rows_rejected_for_compression - Rows rejected for compression |
| REORG_LONG_TBSPC_ID | BIGINT | reorg_long_tbsp_id - Table space where long objects are reorganized |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPUTIL_PROGRESS administrative view and SNAP_GET_UTIL_PROGRESS table function - Retrieve progress logical data group snapshot information

The `SNAPUTIL_PROGRESS` administrative view and the `SNAP_GET_UTIL_PROGRESS` table function return snapshot information about utility progress, in particular, the progress logical data group.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “`SNAPUTIL_PROGRESS` administrative view” on page 961

- “SNAP_GET_UTIL_PROGRESS table function” on page 962

SNAPUTIL_PROGRESS administrative view

Used in conjunction with the SNAPUTIL administrative view, the SNAPUTIL_PROGRESS administrative view provides the same information as the **LIST UTILITIES SHOW DETAIL** CLP command.

The schema is SYSIBMADM.

Refer to Table 196 on page 963 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPUTIL_PROGRESS administrative view
- CONTROL privilege on the SNAPUTIL_PROGRESS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_UTIL_PROGRESS table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve details on total and completed units of progress by utility ID.

```
SELECT UTILITY_ID, PROGRESS_TOTAL_UNITS, PROGRESS_COMPLETED_UNITS,
       DBPARTITIONNUM FROM SYSIBMADM.SNAPUTIL_PROGRESS
```

The following is an example of output from this query.

| UTILITY_ID | PROGRESS_TOTAL_UNITS | PROGRESS_COMPLETED_UNITS | DBPARTITIONNUM |
|------------|----------------------|--------------------------|----------------|
| 7 | 10 | 5 | 0 |
| 9 | 10 | 5 | 1 |

1 record(s) selected.

SNAP_GET_UTIL_PROGRESS table function

The SNAP_GET_UTIL_PROGRESS table function returns the same information as the SNAPUTIL_PROGRESS administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used in conjunction with the SNAP_GET_UTIL table function, the SNAP_GET_UTIL_PROGRESS table function provides the same information as the **LIST UTILITIES SHOW DETAIL CLP** command.

Refer to Table 196 on page 963 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_UTIL_PROGRESS (*member*) ►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_UTIL_PROGRESS table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_UTIL_PROGRESS table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve details on the progress of utilities on the currently connect member.

```
SELECT UTILITY_ID, PROGRESS_TOTAL_UNITS, PROGRESS_COMPLETED_UNITS,  
       DBPARTITIONNUM FROM TABLE(SNAP_GET_UTIL_PROGRESS(-1)) as T
```

The following is an example of output from this query.

```
UTILITY_ID PROGRESS_TOTAL_UNITS PROGRESS_COMPLETED_UNITS DBPARTITIONNUM  
-----  
          7                10                5                0
```

1 record(s) selected.

Information returned

Table 196. Information returned by the SNAPUTIL_PROGRESS administrative view and the SNAP_GET_UTIL_PROGRESS table function

| Column name | Data type | Description or corresponding monitor element |
|----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| UTILITY_ID | INTEGER | utility_id - Utility ID. Unique to a database partition. |
| PROGRESS_SEQ_NUM | INTEGER | progress_seq_num - Progress sequence number. If serial, the number of the phase. If concurrent, then could be NULL. |
| UTILITY_STATE | VARCHAR(16) | utility_state - Utility state. This interface returns a text identifier based on the defines in sqlmon.h |
| PROGRESS_DESCRIPTION | VARCHAR(2048) | progress_description - Progress description |
| PROGRESS_START_TIME | TIMESTAMP | progress_start_time - Progress start time. Start time if the phase has started, otherwise NULL. |
| PROGRESS_WORK_METRIC | VARCHAR(16) | progress_work_metric - Progress work metric. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none">• NOT_SUPPORT• BYTES• EXTENTS• INDEXES• PAGES• ROWS• TABLES |
| PROGRESS_TOTAL_UNITS | BIGINT | progress_total_units - Total progress work units |

Table 196. Information returned by the SNAPUTIL_PROGRESS administrative view and the SNAP_GET_UTIL_PROGRESS table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|-----------|--|
| PROGRESS_COMPLETED_UNITS | BIGINT | progress_completed_units - Completed progress work units |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL table function - Retrieve dynsql logical group snapshot information

The SNAPDYN_SQL administrative view and the SNAP_GET_DYN_SQL table function return snapshot information from the dynsql logical data group.

The “MON_GET_PKG_CACHE_STMT table function - Get package cache statement metrics” on page 686 returns a point-in-time view of both static and dynamic SQL statements in the database package cache. If you use this table function, you can take advantage of the monitoring infrastructure introduced in Version 9.7.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPDYN_SQL administrative view” on page 940
- “SNAP_GET_DYN_SQL table function” on page 942

SNAPDYN_SQL administrative view

This administrative view allows you to retrieve dynsql logical group snapshot information for the currently connected database.

This view returns information equivalent to the **GET SNAPSHOT FOR DYNAMIC SQL ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 192 on page 944 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPDYN_SQL administrative view
- CONTROL privilege on the SNAPDYN_SQL administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following authorizations is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DYN_SQL table function

- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of dynamic SQL run on the currently connected database, ordered by the number of rows read.

```
SELECT PREP_TIME_WORST, NUM_COMPILATIONS, SUBSTR(STMT_TEXT, 1, 60)
      AS STMT_TEXT, DBPARTITIONNUM
  FROM SYSIBMADM.SNAPDYN_SQL ORDER BY ROWS_READ
```

The following is an example of output from this query.

| PREP_TIME_WORST | NUM_COMPILATIONS | ... |
|-----------------|------------------|-----|
| ----- | ----- | ... |
| 98 | 1 | ... |
| 9 | 1 | ... |
| 0 | 0 | ... |
| 0 | 1 | ... |
| 0 | 1 | ... |
| 0 | 1 | ... |
| 0 | 1 | ... |
| 0 | 1 | ... |
| 0 | 1 | ... |
| 40 | 1 | ... |

9 record(s) selected.

Output from this query (continued).

```
... STMT_TEXT
... -----
... select prep_time_worst, num_compilations, substr(stmt_text,
... select * from dbuser.employee
... SET CURRENT LOCALE LC_CTYPE = 'en_US'
... select prep_time_worst, num_compilations, substr(stmt_text,
... select prep_time_worst, num_compilations, substr(stmt_text,
... select * from dbuser.employee
... insert into dbuser.employee values(1)
... select * from dbuser.employee
... insert into dbuser.employee values(1)
```

Output from this query (continued).

```
... DBPARTITIONNUM
... -----
... 0
... 0
... 0
... 2
```

```

...      1
...      2
...      2
...      1
...      0

```

SNAP_GET_DYN_SQL table function

The SNAP_GET_DYN_SQL table function returns the same information as the SNAPDYN_SQL administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

This table function returns information equivalent to the **GET SNAPSHOT FOR DYNAMIC SQL ON database-alias** CLP command.

Refer to Table 192 on page 944 for a complete list of information that can be returned.

Syntax

```

▶▶—SNAP_GET_DYN_SQL—(—dbname—, member)—▶▶

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current member, or -2 for an aggregate of all active members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from members where the database is active.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DYN_SQL table function takes a snapshot for the currently connected database and database member number.

Important: The SYSPROC.SNAP_WRITE_FILE procedure is deprecated in Version 10.5 and might be removed in a future release. For more information, see "SNAP_WRITE_FILE procedure" in *Administrative Routines and Views*.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DYN_SQL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of dynamic SQL run on the currently connected database, ordered by the number of rows read.

```
SELECT PREP_TIME_WORST, NUM_COMPILATIONS, SUBSTR(STMT_TEXT, 1, 60)
      AS STMT_TEXT FROM TABLE(SNAP_GET_DYN_SQL('','-1')) as T
      ORDER BY ROWS_READ
```

The following is an example of output from this query.

| PREP_TIME_WORST | NUM_COMPILATIONS | ... |
|-----------------|------------------|-----|
| 0 | 0 | ... |
| 49 | 1 | ... |
| 0 | 0 | ... |
| 46 | 1 | ... |
| 0 | 0 | ... |
| 0 | 0 | ... |
| 0 | 0 | ... |
| 29 | 1 | ... |
| 0 | 0 | ... |
| 0 | 0 | ... |
| 10 | 1 | ... |
| 0 | 0 | ... |
| 4 | 0 | ... |
| 53 | 0 | ... |
| 0 | 0 | ... |
| 6 | 1 | ... |
| 334 | 0 | ... |
| 0 | 0 | ... |
| 5 | 0 | ... |
| 10 | 0 | ... |
| 599 | 0 | ... |
| 15 | 1 | ... |
| 7 | 0 | ... |

23 record(s) selected.

Output from this query (continued).

```
... STMT_TEXT
...
... SET :HV00017 :HI00017 = RPAD(VARCHAR(:HV00035 :HI00035 ),
```

```

... SELECT COLNAME, TYPENAME FROM SYSCAT.COLUMNS WHERE TABNAME=
... DECLARE RES CURSOR WITH RETURN TO CALLER FOR SELECT R.TEXT F
... SELECT PREP_TIME_WORST, NUM_COMPILATIONS, SUBSTR(STMT_TEXT,
... VALUES (:HV00026 :HI00026 + 1, :HV00024 :HI00024 + 1) IN
... VALUES (:HV00035 :HI00035 + 1, :HV00024 :HI00024 + 1) IN
... VALUES (1) INTO :HV00035 :HI00035
... SELECT TRIGNAME FROM SYSCAT.TRIGGERS WHERE TABNAME='POLICY'
... VALUES (:HV00024 :HI00024 +1, :HV00022 :HI00022 +1) INTO :
... VALUES (1, CARDINALITY(CAST(:HV00040 :HI00040 AS "SYSIBMAD
... CALL SYSPROC.SYSINSTALLOBJECTS('POLICY','V','I','I')
... SET :HV00017 :HI00017 = RPAD(VARCHAR(:HV00035 :HI00035 ),
... drop event monitor act
... SELECT TABSCHEMA, TABNAME, TYPE, STATUS, TBSPACEID, PROPERTY
... CALL SAVE_EXEC_INFO (CAST(:HV00040 :HI00040 AS "SYSIBMADM"
... SET CURRENT LOCK TIMEOUT 5
... SELECT TABNAME FROM SYSCAT.PERIODS WHERE PERIODNAME = 'SYSTE
... SELECT ARRAY_AGG(P.EXECUTABLE_ID ORDER BY M.IO_WAIT_TIME DES
... SET CURRENT ISOLATION RESET
... CALL monreport.pkgcache()
... SELECT A.SPECIFICNAME FROM SYSCAT.ROUTINES A WHERE (A.FENCED
... SELECT POLICY FROM SYSTOOLS.POLICY WHERE MED='DB2CommonMED'
... VALUES 0

```

23 record(s) selected.

After running a workload, user can use the following query with the table function.

```

SELECT STATS_FABRICATE_TIME,SYNC_RUNSTATS_TIME
FROM TABLE (SNAP_GET_DYN_SQL(TmytestdbT, -1))
AS SNAPDB

```

| STATS_FABRICATE_TIME | SYNC_RUNSTATS_TIME |
|----------------------|--------------------|
| 2 | 12 |
| 1 | 30 |

For the view based on this table function:

```

SELECT STATS_FABRICATE_TIME,SYNC_RUNSTATS_TIME
FROM SYSIBMADM.SNAPDYN_SQL

```

| STATS_FABRICATE_TIME | SYNC_RUNSTATS_TIME |
|----------------------|--------------------|
| 5 | 10 |
| 3 | 20 |

2 record(s) selected.

Information returned

Table 197. Information returned by the SNAPDYN_SQL administrative view and the SNAP_GET_DYN_SQL table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| NUM_EXECUTIONS | BIGINT | num_executions - Statement executions |
| NUM_COMPILATIONS | BIGINT | num_compilations - Statement compilations |
| PREP_TIME_WORST | BIGINT | prep_time_worst - Statement worst preparation time |
| PREP_TIME_BEST | BIGINT | prep_time_best - Statement best preparation time |

Table 197. Information returned by the SNAPDYN_SQL administrative view and the SNAP_GET_DYN_SQL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal rows deleted |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal rows inserted |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal rows updated |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| STMT_SORTS | BIGINT | stmt_sorts - Statement sorts |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total sort time |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| TOTAL_EXEC_TIME | BIGINT | total_exec_time - Elapsed statement execution time (in seconds)* |
| TOTAL_EXEC_TIME_MS | BIGINT | total_exec_time - Elapsed statement execution time (fractional, in microseconds)* |
| TOTAL_USR_CPU_TIME | BIGINT | total_usr_cpu_time - Total user CPU for a statement (in seconds)* |
| TOTAL_USR_CPU_TIME_MS | BIGINT | total_usr_cpu_time - Total user CPU for a statement (fractional, in microseconds)* |
| TOTAL_SYS_CPU_TIME | BIGINT | total_sys_cpu_time - Total system CPU for a statement (in seconds)* |

Table 197. Information returned by the SNAPDYN_SQL administrative view and the SNAP_GET_DYN_SQL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--|-----------|--|
| TOTAL_SYS_CPU_TIME_MS | BIGINT | total_sys_cpu_time - Total system CPU for a statement (fractional, in microseconds)* |
| STMT_TEXT | CLOB(2 M) | stmt_text - SQL statement text |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| STATS_FABRICATE_TIME | BIGINT | The total time (in milliseconds) spent by system to create needed statistics without table or index scan during query compilation for a dynamic statement. |
| SYNC_RUNSTATS_TIME | BIGINT | The total time (in milliseconds) spent on synchronous statistics-collect activities during query compilation for a dynamic statement. |
| MEMBER | SMALLINT | member - Database member monitor element |
| <p>* To calculate the total time spent for the monitor element that this column is based on, you must add the full seconds reported in the column for this monitor element that ends with _S to the fractional seconds reported in the column for this monitor element that ends with _MS, using the following formula: $(\text{monitor-element-name_S} \times 1,000,000 + \text{monitor-element-name_MS}) \div 1,000,000$. For example, $(\text{ELAPSED_EXEC_TIME_S} \times 1,000,000 + \text{ELAPSED_EXEC_TIME_MS}) \div 1,000,000$.</p> | | |

SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function - Retrieve subsection logical monitor group snapshot information

The SNAPSUBSECTION administrative view and the SNAP_GET_SUBSECTION table function return information about application subsections, namely the subsection logical monitor grouping.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPSUBSECTION administrative view” on page 946
- “SNAP_GET_SUBSECTION table function” on page 947

SNAPSUBSECTION administrative view

This administrative view allows you to retrieve subsection logical monitor group snapshot information for the currently connected database.

Used with the MON_CURRENT_SQL administrative view and the MON_GET_AGENT, MON_GET_CONNECTION, MON_GET_MEMORY_POOL, and MON_GET_MEMORY_SET table functions, the SNAPSUBSECTION administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS on database-alias** CLP command, but retrieves data from all database members.

The schema is SYSIBMADM.

Refer to Table 193 on page 949 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPSUBSECTION administrative view
- CONTROL privilege on the SNAPSUBSECTION administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_SUBSECTION table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Get status for subsections executing on all database members.

```
SELECT DB_NAME, STMT_TEXT, SS_STATUS, DBPARTITIONNUM
FROM SYSIBMADM.SNAPSUBSECTION
ORDER BY DB_NAME, SS_STATUS, DBPARTITIONNUM
```

The following is an example of output from this query.

| DB_NAME | STMT_TEXT | SS_STATUS | DBPARTITIONNUM |
|---------|------------------------|-----------|----------------|
| SAMPLE | select * from EMPLOYEE | EXEC | 0 |
| SAMPLE | select * from EMPLOYEE | EXEC | 1 |

SNAP_GET_SUBSECTION table function

The SNAP_GET_SUBSECTION table function returns the same information as the SNAPSUBSECTION administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Refer to Table 193 on page 949 for a complete list of information that can be returned.

Used with the SNAP_GET_AGENT, SNAP_GET_AGENT_MEMORY_POOL, SNAP_GET_APPL, SNAP_GET_APPL_INFO and SNAP_GET_STMT table functions, the SNAP_GET_SUBSECTION table function provides information

equivalent to the **GET SNAPSHOT FOR ALL APPLICATIONS** CLP command, but retrieves data from all database members.

Syntax

►► SNAP_GET_SUBSECTION ((*dbname* [, *member*])) ►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_SUBSECTION table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_SUBSECTION table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Get status for subsections executing on all database members.

```
SELECT DB_NAME, STMT_TEXT, SS_STATUS, DBPARTITIONNUM
FROM TABLE(SYSPROC.SNAP_GET_SUBSECTION( '', 0 )) as T
ORDER BY DB_NAME, SS_STATUS, DBPARTITIONNUM
```

The following is an example of output from this query.

| DB_NAME | STMT_TEXT | SS_STATUS | DBPARTITIONNUM |
|---------|------------------------|-----------|----------------|
| SAMPLE | select * from EMPLOYEE | EXEC | 0 |
| SAMPLE | select * from EMPLOYEE | EXEC | 1 |

Information returned

Table 198. Information returned by the SNAPSUBSECTION administrative view and the SNAP_GET_SUBSECTION table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| STMT_TEXT | CLOB(16 M) | stmt_text - SQL statement text |
| SS_EXEC_TIME | BIGINT | ss_exec_time - Subsection execution elapsed time |
| TQ_TOT_SEND_SPILLS | BIGINT | tq_tot_send_spills - Total number of table queue buffers overflowed |
| TQ_CUR_SEND_SPILLS | BIGINT | tq_cur_send_spills - Current number of table queue buffers overflowed |
| TQ_MAX_SEND_SPILLS | BIGINT | tq_max_send_spills - Maximum number of table queue buffers overflows |
| TQ_ROWS_READ | BIGINT | tq_rows_read - Number of rows read from table queues |
| TQ_ROWS_WRITTEN | BIGINT | tq_rows_written - Number of rows written to table queues |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| SS_USR_CPU_TIME_S | BIGINT | ss_usr_cpu_time - User CPU time used by subsection (in seconds)* |
| SS_USR_CPU_TIME_MS | BIGINT | ss_usr_cpu_time - User CPU time used by subsection (fractional, in microseconds)* |
| SS_SYS_CPU_TIME_S | BIGINT | ss_sys_cpu_time - System CPU time used by subsection (in seconds)* |
| SS_SYS_CPU_TIME_MS | BIGINT | ss_sys_cpu_time - System CPU time used by subsection (fractional, in microseconds)* |
| SS_NUMBER | INTEGER | ss_number - Subsection number |

Table 198. Information returned by the SNAPSUBSECTION administrative view and the SNAP_GET_SUBSECTION table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--|-------------|--|
| SS_STATUS | VARCHAR(20) | ss_status - Subsection status. This interface returns a text identifier based on defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • EXEC • TQ_WAIT_TO_RCV • TQ_WAIT_TO_SEND • COMPLETED |
| SS_NODE_NUMBER | SMALLINT | ss_node_number - Subsection node number |
| TQ_NODE_WAITED_FOR | SMALLINT | tq_node_waited_for - Waited for node on a table queue |
| TQ_WAIT_FOR_ANY | INTEGER | tq_wait_for_any - Waiting for any node to send on a table queue |
| TQ_ID_WAITING_ON | INTEGER | tq_id_waiting_on - Waited on node on a table queue |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| <p>* To calculate the total time spent for the monitor element that this column is based on, you must add the full seconds reported in the column for this monitor element that ends with _S to the fractional seconds reported in the column for this monitor element that ends with _MS, using the following formula: $(\text{monitor-element-name_S} \times 1,000,000 + \text{monitor-element-name_MS}) \div 1,000,000$. For example, $(\text{ELAPSED_EXEC_TIME_S} \times 1,000,000 + \text{ELAPSED_EXEC_TIME_MS}) \div 1,000,000$.</p> | | |

SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function - Retrieve database snapshot switch state information

The SNAPSWITCHES administrative view and the SNAP_GET_SWITCHES table function return information about the database snapshot switch state.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPSWITCHES administrative view” on page 950
- “SNAP_GET_SWITCHES table function” on page 952

SNAPSWITCHES administrative view

This view provides the data equivalent to the **GET DBM MONITOR SWITCHES CLP** command.

The schema is SYSIBMADM.

Refer to Table 194 on page 953 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPSWITCHES administrative view
- CONTROL privilege on the SNAPSWITCHES administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_SWITCHES table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve DBM monitor switches state information for all database members.

```
SELECT UOW_SW_STATE, STATEMENT_SW_STATE, TABLE_SW_STATE, BUFFPOOL_SW_STATE,
       LOCK_SW_STATE, SORT_SW_STATE, TIMESTAMP_SW_STATE,
       DBPARTITIONNUM FROM SYSIBMADM.SNAPSWITCHES
```

The following is an example of output from this query.

| UOW_SW_STATE | STATEMENT_SW_STATE | TABLE_SW_STATE | BUFFPOOL_SW_STATE | ... |
|--------------|--------------------|----------------|-------------------|-----|
| 0 | 0 | 0 | 0 | ... |
| 0 | 0 | 0 | 0 | ... |
| 0 | 0 | 0 | 0 | ... |
| ... | | | | |

3 record selected.

Output from this query (continued).

| ... | LOCK_SW_STATE | SORT_SW_STATE | TIMESTAMP_SW_STATE | DBPARTITIONNUM |
|-----|---------------|---------------|--------------------|----------------|
| ... | 1 | 0 | 1 | 0 |
| ... | 1 | 0 | 1 | 1 |
| ... | 1 | 0 | 1 | 2 |

SNAP_GET_SWITCHES table function

The SNAP_GET_SWITCHES table function returns the same information as the SNAPSWITCHES administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

This table function provides the data equivalent to the **GET DBM MONITOR SWITCHES** CLP command.

Refer to Table 194 on page 953 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_SWITCHES (*member*) ►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_SWITCHES table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_SWITCHES table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Retrieve DBM monitor switches state information for the current database member.

```
SELECT UOW_SW_STATE, STATEMENT_SW_STATE, TABLE_SW_STATE,
       BUFFPOOL_SW_STATE, LOCK_SW_STATE, SORT_SW_STATE, TIMESTAMP_SW_STATE
FROM TABLE(SNAP_GET_SWITCHES(-1)) AS T
```

The following is an example of output from this query.

```
UOW_SW_STATE STATEMENT_SW_STATE TABLE_SW_STATE...
-----
1 1 1...
...
1 record(s) selected. ...
```

Output from this query (continued).

```
... BUFFPOOL_SW_STATE LOCK_SW_STATE SORT_SW_STATE TIMESTAMP_SW_STATE
... -----
... 1 1 0 1
```

Information returned

Table 199. Information returned by the SNAPSWITCHES administrative view and the SNAP_GET_SWITCHES table function

| Column name | Data type | Description |
|--------------------|-----------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| UOW_SW_STATE | SMALLINT | State of the unit of work monitor recording switch (0 or 1). |
| UOW_SW_TIME | TIMESTAMP | If the unit of work monitor recording switch is on, the date and time that this switch was turned on. |
| STATEMENT_SW_STATE | SMALLINT | State of the SQL statement monitor recording switch (0 or 1). |
| STATEMENT_SW_TIME | TIMESTAMP | If the SQL statement monitor recording switch is on, the date and time that this switch was turned on. |
| TABLE_SW_STATE | SMALLINT | State of the table activity monitor recording switch (0 or 1). |
| TABLE_SW_TIME | TIMESTAMP | If the table activity monitor recording switch is on, the date and time that this switch was turned on. |
| BUFFPOOL_SW_STATE | SMALLINT | State of the buffer pool activity monitor recording switch (0 or 1). |
| BUFFPOOL_SW_TIME | TIMESTAMP | If the buffer pool activity monitor recording switch is on, the date and time that this switch was turned on. |
| LOCK_SW_STATE | SMALLINT | State of the lock monitor recording switch (0 or 1). |

Table 199. Information returned by the SNAPSWITCHES administrative view and the SNAP_GET_SWITCHES table function (continued)

| Column name | Data type | Description |
|--------------------|-----------|--|
| LOCK_SW_TIME | TIMESTAMP | If the lock monitor recording switch is on, the date and time that this switch was turned on. |
| SORT_SW_STATE | SMALLINT | State of the sorting monitor recording switch (0 or 1). |
| SORT_SW_TIME | TIMESTAMP | If the sorting monitor recording switch is on, the date and time that this switch was turned on. |
| TIMESTAMP_SW_STATE | SMALLINT | State of the timestamp monitor recording switch (0 or 1) |
| TIMESTAMP_SW_TIME | TIMESTAMP | If the timestamp monitor recording switch is on, the date and time that this switch was turned on. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPTAB_REORG administrative view and SNAP_GET_TAB_REORG table function - Retrieve table reorganization snapshot information

The SNAPTAB_REORG administrative view and the SNAP_GET_TAB_REORG table function return table reorganization information.

If no tables have been reorganized, 0 rows are returned. When a data partitioned table is reorganized, one record for each data partition is returned. If only a specific data partition of a data partitioned table is reorganized, only a record for the partition is returned.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPTAB_REORG administrative view” on page 954
- “SNAP_GET_TAB_REORG table function” on page 956

SNAPTAB_REORG administrative view

This administrative view allows you to retrieve table reorganization snapshot information for the currently connected database.

Used with the SNAPTAB administrative view, the SNAPTAB_REORG administrative view provides the data equivalent to the **GET SNAPSHOT FOR TABLES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 195 on page 958 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPTAB_REORG administrative view
- CONTROL privilege on the SNAPTAB_REORG administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_TAB_REORG table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Select details on reorganization operations for all database members on the currently connected database.

```
SELECT SUBSTR(TABNAME, 1, 15) AS TAB_NAME, SUBSTR(TABSCHEMA, 1, 15)
      AS TAB_SCHEMA, REORG_PHASE, SUBSTR(REORG_TYPE, 1, 20) AS REORG_TYPE,
      REORG_STATUS, REORG_COMPLETION, DBPARTITIONNUM
FROM SYSIBMADM.SNAPTAB_REORG ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| TAB_NAME | TAB_SCHEMA | REORG_PHASE | ... |
|----------|------------|-------------|-----|
| EMPLOYEE | DBUSER | REPLACE | ... |
| EMPLOYEE | DBUSER | REPLACE | ... |
| EMPLOYEE | DBUSER | REPLACE | ... |

3 record(s) selected.

Output from this query (continued).

| ... REORG_TYPE | REORG_STATUS | REORG_COMPLETION | DBPARTITIONNUM |
|--------------------------|--------------|------------------|----------------|
| ... RECLAIM+OFFLINE+ALLO | COMPLETED | SUCCESS | 0 |
| ... RECLAIM+OFFLINE+ALLO | COMPLETED | SUCCESS | 1 |
| ... RECLAIM+OFFLINE+ALLO | COMPLETED | SUCCESS | 2 |

Select all information about a reorganization operation to reclaim extents from a multidimensional clustering (MDC) or insert time clustering (ITC) table from the SNAPTAB_REORG administrative view.

```
db2 -v "select * from sysibmadm.snaptab_reorg"
```

| TABNAME | REORG_PHASE | REORG_MAX_PHASE | REORG_TYPE |
|--------------|------------------|----------------------------|-----------------------------|
| T1 | RELEASE | 3 | RECLAIM_EXTENTS+ALLOW_WRITE |
| REORG_STATUS | REORG_COMPLETION | REORG_START | REORG_END |
| COMPLETED | SUCCESS | 2008-09-24-14.35.30.734741 | 2008-09-24-14.35.31.460674 |

SNAP_GET_TAB_REORG table function

The SNAP_GET_TAB_REORG table function returns the same information as the SNAPTAB_REORG administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_TAB table function, the SNAP_GET_TAB_REORG table function provides the data equivalent to the **GET SNAPSHOT FOR TABLES ON database-alias** CLP command.

Refer to Table 195 on page 958 for a complete list of information that can be returned.

Syntax

```

>> SNAP_GET_TAB_REORG ( ( dbname [ , member ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TAB_REORG table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_TAB_REORG table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Select details on reorganization operations for database member 1 on the currently connected database.

```
SELECT SUBSTR(TABNAME, 1, 15) AS TAB_NAME, SUBSTR(TABSCHEMA, 1, 15)
      AS TAB_SCHEMA, REORG_PHASE, SUBSTR(REORG_TYPE, 1, 20) AS REORG_TYPE,
      REORG_STATUS, REORG_COMPLETION, DBPARTITIONNUM
FROM TABLE( SNAP_GET_TAB_REORG('', 1)) AS T
```

The following is an example of output from this query.

| TAB_NAME | TAB_SCHEMA | REORG_PHASE | REORG_TYPE | ... |
|-----------------------|------------|-------------|----------------------|-----|
| EMPLOYEE | DBUSER | REPLACE | RECLAIM+OFFLINE+ALLO | ... |
| 1 record(s) selected. | | | | ... |

Output from this query (continued).

| ... | REORG_STATUS | REORG_COMPLETION | DBPARTITIONNUM |
|-----|--------------|------------------|----------------|
| ... | COMPLETED | SUCCESS | 1 |
| ... | | | |

Select all information about a reorganization operation to reclaim extents from a multidimensional clustering (MDC) or insert time clustering (ITC) table using the SNAP_GET_TAB_REORG table function.

```
db2 -v "select * from table(snap_get_tab_reorg(''))"
```

| TABNAME | REORG_PHASE | REORG_MAX_PHASE | REORG_TYPE |
|---------|-------------|-----------------|-----------------------------|
| T1 | RELEASE | 3 | RECLAIM_EXTENTS+ALLOW_WRITE |

| REORG_STATUS | REORG_COMPLETION | REORG_START | REORG_END |
|--------------|------------------|----------------------------|----------------------------|
| COMPLETED | SUCCESS | 2008-09-24-14.35.30.734741 | 2008-09-24-14.35.31.460674 |

Information returned

Table 200. Information returned by the *SNAPTAB_REORG* administrative view and the *SNAP_GET_TAB_REORG* table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABNAME | VARCHAR (128) | table_name - Table name |
| TABSCHEMA | VARCHAR (128) | table_schema - Table schema name |
| PAGE_REORGs | BIGINT | page_reorgs - Page reorganizations |
| REORG_PHASE | VARCHAR (16) | reorg_phase - Table reorganize phase. This interface returns a text identifier based on defines in <i>sqlmon.h</i> and is one of: <ul style="list-style-type: none">• BUILD• DICT_SAMPLE• INDEX_RECREATE• REPLACE• SORT• SCAN• DRAIN• RELEASE or SORT+DICT_SAMPLE. |
| REORG_MAX_PHASE | INTEGER | reorg_max_phase - Maximum table reorganize phase |
| REORG_CURRENT_COUNTER | BIGINT | reorg_current_counter - Table reorganize progress |
| REORG_MAX_COUNTER | BIGINT | reorg_max_counter - Total amount of table reorganization |

Table 200. Information returned by the SNAPTAB_REORG administrative view and the SNAP_GET_TAB_REORG table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------|------------------|---|
| REORG_TYPE | VARCHAR
(128) | <p>reorg_type - Table reorganize attributes. This interface returns a text identifier using a combination of the following identifiers separated by '+':</p> <p>Either:</p> <ul style="list-style-type: none"> • RECLAIM • RECLUSTER • RECLAIM_EXTS <p>and either:</p> <ul style="list-style-type: none"> • +OFFLINE • +ONLINE <p>If access mode is specified, it is one of:</p> <ul style="list-style-type: none"> • +ALLOW_NONE • +ALLOW_READ • +ALLOW_WRITE <p>If offline and RECLUSTER option, one of:</p> <ul style="list-style-type: none"> • +INDEXSCAN • +TABLESCAN <p>If offline, one of:</p> <ul style="list-style-type: none"> • +LONGLOB • +DATAONLY <p>If offline, and option is specified, any of:</p> <ul style="list-style-type: none"> • +CHOOSE_TEMP • +KEEPDICTIONARY • +RESETDICTIONARY <p>If online, and option is specified:</p> <ul style="list-style-type: none"> • +NOTRUNCATE <p>Example 1: If a REORG TABLE TEST.EMPLOYEE was run, the following would be displayed:</p> <p>RECLAIM+OFFLINE+ALLOW_READ+DATAONLY+KEEPDICTIONARY</p> <p>Example 2: If a REORG TABLE TEST.EMPLOYEE INDEX EMPIDX INDEXSCAN was run, then the following would be displayed:</p> <p>RECLUSTER+OFFLINE+ALLOW_READ+INDEXSCAN+DATAONLY+KEEPDICTIONARY</p> |

Table 200. Information returned by the `SNAPTAB_REORG` administrative view and the `SNAP_GET_TAB_REORG` table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|--------------|---|
| REORG_STATUS | VARCHAR (10) | reorg_status - Table reorganize status. This interface returns a text identifier based on defines in <code>sqlmon.h</code> and is one of: <ul style="list-style-type: none"> • COMPLETED • PAUSED • STARTED • STOPPED • TRUNCATE |
| REORG_COMPLETION | VARCHAR (10) | reorg_completion - Table reorganization completion flag. This interface returns a text identifier, based on defines in <code>sqlmon.h</code> and is one of: <ul style="list-style-type: none"> • FAIL • SUCCESS |
| REORG_START | TIMESTAMP | reorg_start - Table reorganize start time |
| REORG_END | TIMESTAMP | reorg_end - Table reorganize end time |
| REORG_PHASE_START | TIMESTAMP | reorg_phase_start - Table reorganize phase start time |
| REORG_INDEX_ID | BIGINT | reorg_index_id - Index used to reorganize the table |
| REORG_TBSPC_ID | BIGINT | reorg_tbspc_id - Table space where table is reorganized |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition identifier. For a non-partitioned table, this element will be NULL. |
| REORG_ROWSCOMPRESSED | BIGINT | reorg_rows_compressed - Rows compressed |
| REORG_ROWSREJECTED | BIGINT | reorg_rows_rejected_for_compression - Rows rejected for compression |
| REORG_LONG_TBSPC_ID | BIGINT | reorg_long_tbspc_id - Table space where long objects are reorganized |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPUTIL_PROGRESS administrative view and SNAP_GET_UTIL_PROGRESS table function - Retrieve progress logical data group snapshot information

The `SNAPUTIL_PROGRESS` administrative view and the `SNAP_GET_UTIL_PROGRESS` table function return snapshot information about utility progress, in particular, the progress logical data group.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “`SNAPUTIL_PROGRESS` administrative view” on page 961

- “SNAP_GET_UTIL_PROGRESS table function” on page 962

SNAPUTIL_PROGRESS administrative view

Used in conjunction with the SNAPUTIL administrative view, the SNAPUTIL_PROGRESS administrative view provides the same information as the **LIST UTILITIES SHOW DETAIL** CLP command.

The schema is SYSIBMADM.

Refer to Table 196 on page 963 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPUTIL_PROGRESS administrative view
- CONTROL privilege on the SNAPUTIL_PROGRESS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_UTIL_PROGRESS table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve details on total and completed units of progress by utility ID.

```
SELECT UTILITY_ID, PROGRESS_TOTAL_UNITS, PROGRESS_COMPLETED_UNITS,
       DBPARTITIONNUM FROM SYSIBMADM.SNAPUTIL_PROGRESS
```

The following is an example of output from this query.

| UTILITY_ID | PROGRESS_TOTAL_UNITS | PROGRESS_COMPLETED_UNITS | DBPARTITIONNUM |
|------------|----------------------|--------------------------|----------------|
| 7 | 10 | 5 | 0 |
| 9 | 10 | 5 | 1 |

1 record(s) selected.

SNAP_GET_UTIL_PROGRESS table function

The SNAP_GET_UTIL_PROGRESS table function returns the same information as the SNAPUTIL_PROGRESS administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used in conjunction with the SNAP_GET_UTIL table function, the SNAP_GET_UTIL_PROGRESS table function provides the same information as the **LIST UTILITIES SHOW DETAIL CLP** command.

Refer to Table 196 on page 963 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_UTIL_PROGRESS (member) ►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_UTIL_PROGRESS table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_UTIL_PROGRESS table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve details on the progress of utilities on the currently connect member.

```
SELECT UTILITY_ID, PROGRESS_TOTAL_UNITS, PROGRESS_COMPLETED_UNITS,  
       DBPARTITIONNUM FROM TABLE(SNAP_GET_UTIL_PROGRESS(-1)) as T
```

The following is an example of output from this query.

```
UTILITY_ID PROGRESS_TOTAL_UNITS PROGRESS_COMPLETED_UNITS DBPARTITIONNUM  
-----  
          7                10                5                0
```

1 record(s) selected.

Information returned

Table 201. Information returned by the SNAPUTIL_PROGRESS administrative view and the SNAP_GET_UTIL_PROGRESS table function

| Column name | Data type | Description or corresponding monitor element |
|----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| UTILITY_ID | INTEGER | utility_id - Utility ID. Unique to a database partition. |
| PROGRESS_SEQ_NUM | INTEGER | progress_seq_num - Progress sequence number. If serial, the number of the phase. If concurrent, then could be NULL. |
| UTILITY_STATE | VARCHAR(16) | utility_state - Utility state. This interface returns a text identifier based on the defines in sqlmon.h |
| PROGRESS_DESCRIPTION | VARCHAR(2048) | progress_description - Progress description |
| PROGRESS_START_TIME | TIMESTAMP | progress_start_time - Progress start time. Start time if the phase has started, otherwise NULL. |
| PROGRESS_WORK_METRIC | VARCHAR(16) | progress_work_metric - Progress work metric. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none">• NOT_SUPPORT• BYTES• EXTENTS• INDEXES• PAGES• ROWS• TABLES |
| PROGRESS_TOTAL_UNITS | BIGINT | progress_total_units - Total progress work units |

Table 201. Information returned by the SNAPUTIL_PROGRESS administrative view and the SNAP_GET_UTIL_PROGRESS table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|-----------|--|
| PROGRESS_COMPLETED_UNITS | BIGINT | progress_completed_units - Completed progress work units |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

TOP_DYNAMIC_SQL administrative view - Retrieve information about the top dynamic SQL statements

The TOP_DYNAMIC_SQL administrative view returns the top dynamic SQL statements sortable by number of executions, average execution time, number of sorts, or sorts per statement.

The queries returned by TOP_DYNAMIC_SQL administrative view are the queries that should get focus to ensure they are well tuned.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the TOP_DYNAMIC_SQL administrative view
- CONTROL privilege on the TOP_DYNAMIC_SQL administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMOINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Information returned

Table 202. Information returned by the TOP_DYNAMIC_SQL administrative view

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | Timestamp for the report. |
| NUM_EXECUTIONS | BIGINT | num_executions - Statement executions |

Table 202. Information returned by the TOP_DYNAMIC_SQL administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|-----------|--|
| AVERAGE_EXECUTION_TIME_S | BIGINT | Average execution time, in seconds. |
| STMT_SORTS | BIGINT | stmt_sorts - Statement sorts |
| SORTS_PER_EXECUTION | BIGINT | Number of sorts per statement execution. |
| STMT_TEXT | CLOB(2 M) | stmt_text - SQL statement text |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

Example

Identify the top 5 most frequently run SQL.

```
SELECT NUM_EXECUTIONS, AVERAGE_EXECUTION_TIME_S, STMT_SORTS,
       SORTS_PER_EXECUTION, SUBSTR(STMT_TEXT,1,60) AS STMT_TEXT
FROM SYSIBMADM.TOP_DYNAMIC_SQL
ORDER BY NUM_EXECUTIONS DESC FETCH FIRST 5 ROWS ONLY
```

The following is an example of output for this query.

| NUM_EXECUTIONS | AVERAGE_EXECUTION_TIME_S | STMT_SORTS | ... |
|----------------|--------------------------|------------|-----|
| 148 | 0 | 0 | ... |
| 123 | 0 | 0 | ... |
| 2 | 0 | 0 | ... |
| 1 | 0 | 0 | ... |
| 1 | 0 | 0 | ... |

5 record(s) selected.

Output for this query (continued).

```
... SORTS_PER_EXECUTION ...
... ----- ...
... 0 ...
... 0 ...
... 0 ...
... 0 ...
... 0 ...
```

Output for this query (continued).

```
... STMT_TEXT
... -----
... SELECT A.ID, B.EMPNO, B.FIRSTNME, B.LASTNAME, A.DEPT FROM E
... SELECT A.EMPNO, A.FIRSTNME, A.LASTNAME, B.LOCATION, B.MGRNO
... SELECT A.EMPNO, A.FIRSTNME, A.LASTNAME, B.DEPTNAME FROM EMP
... SELECT ATM.SCHEMA, ATM.NAME, ATM.CREATE_TIME, ATM.LAST_WAIT,
... SELECT * FROM JESSICAE.EMP_RESUME
```

SQL procedure routines

This grouping of routines and views can be used to interact with SQL routines and packages.

ALTER_ROUTINE_PACKAGE procedure

This procedure alters values for the package associated with a compiled SQL routine or a compiled trigger, without the need for rebinding.

It is functionally equivalent to the ALTER PACKAGE statement, except that it takes an object name instead of a package name as an argument. The ALTER_ROUTINE_PACKAGE procedure can be invoked from the command line or called from an application.

►►ALTER_ROUTINE_PACKAGE(—*type*—,—*schema*—,—*module*—,—*name*—,—*options*—)————►◄

The schema is SYSPROC.

Procedure parameters

type

An input argument of type CHAR(2) that specifies the type of routine or compiled trigger, using one of the following values:

- 'P ' for a procedure
- 'SP' for the specific name of a procedure
- 'F' for a compiled function
- 'SF' for a specific name of a compiled function
- 'T' for a compiled trigger

schema

An optional input argument of type VARCHAR(128), which specifies the schema of the routine or trigger. If a schema is not specified, the value will default to the value of the CURRENT SCHEMA special register. This parameter is case sensitive.

module

An optional input argument of type VARCHAR(128), which specifies the name of the module where the routine resides. This parameter cannot be specified for triggers. If this parameter is not specified, then module routines are ignored. This parameter is case sensitive.

name

An input argument of type VARCHAR(128), which specifies the name of the routine or trigger. This parameter is case sensitive.

options

An input argument of type VARCHAR(1024), which specifies a list of any options supported by the ALTER PACKAGE statement. At least one ALTER PACKAGE clause must be supplied within the *options* parameter.

Authorization

One of the following authorities is required to execute the procedure:

- EXECUTE privilege on the procedure
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Examples

Alter the underlying package for an existing stored procedure by the name of UPDATE_EMPLOYEE.

```
CALL SYSPROC.ALTER_ROUTINE_PACKAGE ('P','','','UPDATE_EMPLOYEE',  
    'ACCESS PLAN REUSE YES OPTIMIZATION PROFILE AYYANG.INDEXHINTS')
```

Alter the package for a compiled trigger called MIN_SALARY, in the DRICARD schema.

```
CALL SYSPROC.ALTER_ROUTINE_PACKAGE ('T','DRICARD','','MIN_SALARY',  
    'OPTIMIZATION PROFILE AYYANG.INDEXHINTS')
```

Alter the package for a compiled function, using a three part name.

```
CALL SYSPROC.ALTER_ROUTINE_PACKAGE ('F','DRICARD','MODULE','FUNCTION','APREUSE YES')
```

GET_ROUTINE_NAME procedure - get SQL PL object name

The GET_ROUTINE_NAME procedure returns the name of an SQL PL object given the object's LIB_ID.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine
- DATAACCESS authority
- SQLADM authority
- DBADM authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the procedure is automatically created.

Syntax

```
►► GET_ROUTINE_NAME (—lib_id—, —type—, —schema—, ——————►  
► —module—, —name—, —specific_name—) ——————►◄
```

The schema is SYSPROC.

Routine parameters

lib_id

An input argument of type INTEGER that specifies the LIB_ID corresponding to a compiled SQL PL routine or trigger. This is the value of the SQLERRD(4) field in the SQLCA for a runtime SQL PL error.

type

An output argument of type CHAR(1) that specifies the type of object that raised the error, using one of the following values:

- 'P' for a procedure
- 'F' for a function
- 'T' for a trigger

schema

An output argument of type VARCHAR(128) that specifies the schema to which the object belongs.

module

An output argument of type VARCHAR(128) that specifies the module to which the object belongs. NULL if the object is a trigger or is not part of a module.

name

An output argument of type VARCHAR(128) that specifies the name of the object.

specific_name

An output argument of type VARCHAR(128) that specifies the specific name of the object. NULL if the object is a trigger.

Usage notes

- When the SQLCA structure contains an error raised during the execution of an SQL PL routine or trigger, the SQLCAID field has the value 'SQLCAM' and SQLERRM(4) has a numeric value that uniquely identifies the object that raised the error. This unique identifier corresponds to the LIB_ID column in the SYSCAT.ROUTINES and SYSCAT.TRIGGERS catalog views. The GET_ROUTINE_NAME routine maps a given LIB_ID value to the corresponding object name.
- If the value provided in the first argument to SYSPROC.GET_ROUTINE_NAME does not correspond to an SQL PL routine or trigger, the procedure returns NULL in all the output parameters, indicating that the routine or trigger could not be found.

Example

Assume the execution of an application finishes with an error described in the following SQLCA structure:

SQLCA Information

```
sqlcaid : SQLCAM      sqlcabc: 136      sqlcode: -801
...
sqlerrd : (1) 0        (2) 0            (3) 3
          (4) 13152254 (5) 0            (6) 0
...
sqlstate: 22012
```

The presence of SQLCAM in the SQLCAID field indicates that the error occurred during the execution of an SQL PL object. The values of 3 and 13152254 in the SQLERRD(3) and SQLERRD(4) fields indicate that the error originated in the statement at line number 3 of the SQL PL object identified by the value 13152254. To determine what the SQL PL object is, call the SYSPROC.GET_ROUTINE_NAME in the following manner:

```
db2 call "GET_ROUTINE_NAME(13152254, ?, ?, ?, ?, ?)"
```

Value of output parameters

Parameter Name : OBJTYPE

```

Parameter Value : P

Parameter Name  : OBJSCHEMA
Parameter Value : APPDEV

Parameter Name  : OBJMODULE
Parameter Value : -

Parameter Name  : OBJNAME
Parameter Value : PROC2

Parameter Name  : OBJSPECIFIC
Parameter Value : APPDEV_PROC2

Return Status = 0

```

The values of the OUT parameters from the call to SYSPROC.GET_ROUTINE_NAME indicate that the error was raised from the stored procedure APPDEV.PROC2.

GET_ROUTINE_OPTS

The GET_ROUTINE_OPTS function returns a character string value of the options that are to be used for the creation of SQL procedures in the current session.

►► GET_ROUTINE_OPTS(—) ————— ◀◀

The schema is SYSPROC.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Information returned

The result of the function is a varying-length character string (VARCHAR) value with a length attribute of 1024.

Example

Return the options to be used for the creation of SQL procedures as the result of a query.

```

SELECT GET_ROUTINE_OPTS()
FROM SYSIBM.SYSDUMMY1

```

GET_ROUTINE_SAR

The GET_ROUTINE_SAR procedure retrieves the necessary information to install the same routine in another database server running the same level on the same operating system.

Important: The GET_ROUTINE_SAR procedure is deprecated, use CREATE PROCEDURE to copy a procedure to another database.

The information retrieved by GET_ROUTINE_SAR is placed into a single BLOB string representing an SQL archive file.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Syntax

```
►►—GET_ROUTINE_SAR—————►
►—(—sarblob—,—type—,—routine-name-string—,—hide-body-flag—)————►
```

The schema is SYSFUN.

Procedure parameters

sarblob

An output argument of type BLOB(3M) that contains the routine SAR file contents.

type

An input argument of type CHAR(2) that specifies the type of routine, using one of the following values:

- 'P ' for a procedure
- 'SP' for the specific name of a procedure

routine-name-string

An input argument of type VARCHAR(257) that specifies a qualified name of the routine. If no schema name is specified, the default is the CURRENT SCHEMA when the routine is processed. The *routine-name-string* cannot include double quotation marks (").

hide-body-flag

An input argument of type INTEGER that specifies (using one of the following values) whether or not the routine body should be hidden when the routine text is extracted from the catalogs. Valid values are:

- 0 Leave the routine text intact. This is the default value.
- 1 Replace the routine body with an empty body when the routine text is extracted from the catalogs.

The qualified name of the routine is used to determine which routine to retrieve. The routine that is found must be an SQL routine. Not using a specific name may result in more than one routine, and an error is raised (SQLSTATE 42725). If this occurs, the specific name of the required routine must be used.

The SAR file must include a bind file, which may not be available at the server. If the bind file cannot be found and stored in the SAR file, an error is raised (SQLSTATE 55045).

PUT_ROUTINE_SAR

The PUT_ROUTINE_SAR procedure passes the necessary file to create an SQL routine at the server and then defines the routine.

Important: The PUT_ROUTINE_SAR procedure is deprecated, use CREATE PROCEDURE to copy a procedure to another database.

Authorization

One of the following authorizations is required to execute the procedure:

- EXECUTE privilege on the procedure
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

```

▶▶ PUT_ROUTINE_SAR (—sarblob— [—new-owner—, —use-register-flag—]) ▶▶

```

The schema is SYSFUN.

Procedure parameters

sarblob

An input argument of type BLOB(3M) that contains the routine SAR file contents.

new-owner

An input argument of type VARCHAR(128) that contains an authorization-name used for authorization checking of the routine. The *new-owner* must have the necessary privileges for the routine to be defined. If *new-owner* is not specified, the authorization-name of the original routine definer is used.

use-register-flag

An input argument of type INTEGER that indicates whether or not the CURRENT SCHEMA and CURRENT PATH special registers are used to define the routine. If the special registers are not used, the settings for the default

schema and SQL path are the settings used when the routine was originally defined. Possible values for *use-register-flag*:

- 0 Do not use the special registers of the current environment
- 1 Use the CURRENT SCHEMA and CURRENT PATH special registers.

If the value is 1, CURRENT SCHEMA is used for unqualified object names in the routine definition (including the name of the routine) and CURRENT PATH is used to resolve unqualified routines and data types in the routine definition. If the *use-registers-flag* is not specified, the behavior is the same as if a value of 0 was specified.

The identification information contained in *sarblob* is checked to confirm that the inputs are appropriate for the environment, otherwise an error is raised (SQLSTATE 55046). The PUT_ROUTINE_SAR procedure then uses the contents of the *sarblob* to define the routine at the server.

The contents of the *sarblob* argument are extracted into the separate files that make up the SQL archive file. The shared library and bind files are written to files in a temporary directory. The environment is set so that the routine definition statement processing is aware that compiling and linking are not required, and that the location of the shared library and bind files is available. The contents of the DDL file are then used to dynamically execute the routine definition statement.

No more than one procedure can be concurrently installed under a given schema.

Processing of this statement might result in the same errors as executing the routine definition statement using other interfaces. During routine definition processing, the presence of the shared library and bind files is noted and the precompile, compile and link steps are skipped. The bind file is used during bind processing and the contents of both files are copied to the usual directory for an SQL routine.

If a GET ROUTINE or a PUT ROUTINE operation (or their corresponding procedure) fails to execute successfully, it will always return an error (SQLSTATE 38000), along with diagnostic text providing information about the cause of the failure. For example, if the procedure name provided to GET ROUTINE does not identify an SQL procedure, diagnostic "-204, 42704" text will be returned, where "-204" is the SQLCODE and "42704" is the SQLSTATE, that identify the cause of the problem. The SQLCODE and SQLSTATE in this example indicate that the procedure name provided in the GET ROUTINE command is undefined.

REBIND_ROUTINE_PACKAGE procedure - rebind a package

The REBIND_ROUTINE_PACKAGE procedure rebinds the package associated with an SQL procedure, routine, compiled function, or trigger.

The REBIND_ROUTINE_PACKAGE procedure is functionally equivalent to the REBIND command, except that it takes a procedure name, instead of a package name, as an argument.

The REBIND_ROUTINE_PACKAGE procedure can be invoked from the command line or called from an application.

Explicitly rebinding the associated package does not revalidate the routine. Revalidate an invalid routine by using automatic revalidation or explicitly by using

the ADMIN_REVALIDATE_DB_OBJECTS procedure. Routine revalidation automatically rebinds the dependent package.

Authorization

One of the following authorities is required to execute the procedure:

- EXECUTE privilege on the procedure
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Syntax

There are two equally valid methods to invoke REBIND_ROUTINE_PACKAGE. The only difference between the two invocations is the method of specifying the routine name. In the first instance, the *routine-name-string* variable consists of identifier names separated by periods. In the second method, the routine is identified by separate values for each of the *schema*, *module* and *name* values.

Method 1:

```
►►REBIND_ROUTINE_PACKAGE—(—type—,—routine-name-string—,—options—)————◄◄
```

Method 2:

```
►►REBIND_ROUTINE_PACKAGE—(—type—,——————►  
►—schema—,—module—,—name—,—options—)————◄◄
```

The schema is SYSPROC.

Procedure parameters

type

An input argument of type CHAR(2) that specifies the type of routine or compiled trigger, using one of the following values:

- 'P ' for a procedure
- 'SP' for the specific name of a procedure
- 'F' for a compiled function
- 'SF' for a specific name of a compiled function
- 'T' for a compiled trigger

routine-name-string (method 1 only)

An input argument of type VARCHAR(386) which specifies the name of the routine or trigger. Trigger names consist of two parts separated by a period and are in the format *schema.trigger* where the *schema* is optional. Routine names consist of three part names separated by periods and are in the format *schema.module.routine* where *schema* and *module* are optional. If *schema* is

not specified, the value defaults to the value of the CURRENT SCHEMA special register. If a two-part name is specified, the first part is initially interpreted as a schema name; if the routine is not found under that schema, the first part is interpreted as a module name, and an attempt is made to find the routine in a module of that name under the CURRENT SCHEMA. The schema, module or object names cannot include double quotation marks (") or periods(.).

schema (method 2 only)

An optional input argument of type VARCHAR(128) that specifies the schema of the routine or trigger. If a schema is not specified, the value will default to the value of the CURRENT SCHEMA special register. This parameter is case sensitive.

module (method 2 only)

An optional input argument of type VARCHAR(128) that specifies the name of the module where the routine resides. Do not specify this parameter for triggers. Module routines are ignored if this parameter is not specified. This parameter is case sensitive.

name (method 2 only)

An input argument of type VARCHAR(128) that specifies the name of the routine or trigger. This parameter is case sensitive.

options

An optional input argument of type VARCHAR(1024) which specifies any list of rebind options following the REBIND command syntax. A single value of "ANY" or "CONSERVATIVE" is also supported for backward compatibility and is interpreted as the value for the RESOLVE rebind option.

The qualified name of the routine is used to determine which routine to retrieve. The routine that is found must be an SQL routine; otherwise, an error is returned (SQLSTATE 428F7). If a specific name is not used, more than one routine may be found, and an error is returned (SQLSTATE 42725). If this occurs, the specific name of the required routine must be used.

Examples

Example 1: Rebind the package of routine UPDATE_EMPLOYEE using the RESOLVE, REOPT and APREUSE options.

```
Method 1:
CALL SYSPROC.REBIND_ROUTINE_PACKAGE (
  'P','UPDATE_EMPLOYEE','RESOLVE ANY REOPT ONCE APREUSE YES')
Method 2:
CALL SYSPROC.REBIND_ROUTINE_PACKAGE (
  'P','','','UPDATE_EMPLOYEE','RESOLVE ANY REOPT ONCE APREUSE YES')
```

Example 2: Rebind the package of routine UPDATE_EMPLOYEE with no options.

```
Method 1:
CALL SYSPROC.REBIND_ROUTINE_PACKAGE (
  'P','UPDATE_EMPLOYEE','')
Method 2:
CALL SYSPROC.REBIND_ROUTINE_PACKAGE (
  'P','','','UPDATE_EMPLOYEE','')
```

Example 3: Rebind the package of a compiled trigger.

```
Method 1:
CALL SYSPROC.REBIND_ROUTINE_PACKAGE (
  'T','DRICARD.MIN_SALARY','REOPT ALWAYS')
```

```
Method 2:
CALL SYSPROC.REBIND_ROUTINE_PACKAGE (
  'T','DRICARD','','MIN_SALARY','REOPT ALWAYS')
```

Example 4: Rebind the package of a compiled function using a three part name.

```
Method 1
CALL SYSPROC.REBIND_ROUTINE_PACKAGE (
  'F','DRICARD.MODULE.FUNCTION','REOPT ALWAYS')
Method 2
CALL SYSPROC.REBIND_ROUTINE_PACKAGE (
  'F','DRICARD','MODULE','FUNCTION','REOPT ALWAYS')
```

SET_ROUTINE_OPTS

The SET_ROUTINE_OPTS procedure sets the options that are to be used for the creation of SQL procedures in the current session.

Options set with the SET_ROUTINE_OPTS procedure override the instance-wide setting specified in the **DB2_SQLROUTINE_PREPOPTS** registry variable.

The schema is SYSPROC.

Authorization

One of the following authorities is required to execute the procedure:

- EXECUTE privilege on the procedure
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Syntax

```
➤—SET_ROUTINE_OPTS—(—character-expression—)—————➤
```

Procedure parameter

character-expression

An input argument of type VARCHAR(1024) that specifies the options setting for the current session.

Specified options are valid for the duration of the session. If the null value is specified as the argument, the value of the **DB2_SQLROUTINE_PREPOPTS** registry variable is restored as the default options setting for the current session. For a list of the allowed options, see the description of the **DB2_SQLROUTINE_PREPOPTS** registry variable under “Query compiler variables”.

Example 1: Set the options setting for the current session to NULL.

```
CALL SYSPROC.SET_ROUTINE_OPTS(CAST (NULL AS VARCHAR(1)))
```

Example 2: Set the options setting for the current session to EXPLAIN YES.

```
CALL SET_ROUTINE_OPTS('EXPLAIN YES')
```

Example 3: Set the options setting for the current session to EXPLAIN YES and BLOCKING NO.

```
CALL SET_ROUTINE_OPTS('EXPLAIN YES BLOCKING NO')
```

Stepwise redistribute routines

This grouping of routines and views can be used to manage data distribution in partitioned database environments.

ANALYZE_LOG_SPACE procedure - Retrieve log space analysis information

The ANALYZE_LOG_SPACE procedure returns the log space analysis results for each of the database partitions of the given database partition group.

Syntax

```
►►—ANALYZE_LOG_SPACE—(—inDBPGroup—,—inMainTbSchema—,—inMainTable—,——————►  
►—analysisType—,—inStmgTime—,—addDropOption—,—addDropList—,—pNumber—,—————►  
►—pWeight—)——————►►
```

The schema is SYSPROC.

Procedure parameters

inDBPGroup

An input argument of type VARCHAR (128) that specifies the database partition group name.

inMainTbSchema

An input argument of type VARCHAR (128) that specifies the schema of the main table

inMainTable

An input argument of type VARCHAR (128) that specifies the main table within the database partition group, usually the largest table in the database partition group.

analysisType

An input argument of type SMALLINT that specifies an indicator for analysis type:

- SWRD_USE_STMG_TABLE (1): indicates that the information in the storage management tables is used to find the table row count per database partition. This type should only be used if the storage management tables are setup, and at least one storage snapshot has been taken for the database partition group that is to be redistributed.
- SWRD_USE_REALTIME_ANALYSIS (2): indicates that a SELECT query is used to find the table row count per database partition.

inStmgTime

An input argument of type VARCHAR (26) that specifies the timestamp for the storage management record. This parameter is ignored when *analysisType* is set to SWRD_USE_REALTIME_ANALYSIS.

addDropOption

An input argument of type CHAR (1) that specifies database partitions are being added or dropped:

- 'A': Adding database partitions.
- 'D': Dropping database partitions.
- 'N': No adding or dropping.

addDropList

An input argument of type VARCHAR (6000) that specifies the database partitions to be added or dropped. This database partition numbers are specified in a comma-separated string format and no spaces are allowed in the string.

pNumber

An input argument of type VARCHAR (6000) that specifies all the database partition numbers corresponding to the database partition weight. Each database partition number is between 0 and 999, and the database partition numbers are specified in a comma-separated string with no spaces in the string.

pWeight

An input argument of type VARCHAR (6000) that specifies all the database partition weights that the user has specified corresponding to the database partition numbers in the *pNumber* string. Each database partition weight is a number between 0 and 32767, and database partition weights are specified in a comma-separated string with no spaces in the string.

Authorization

- SYSADM, SYSMON, SYSCtrl, or SYSMAINT
- EXECUTE privilege on the ANALYZE_LOG_SPACE procedure

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

Analyze the effect of adding a database partition without applying the changes. In the following case, the hypothesis is adding database partition 40, 50 and 60 to the database partition group, and for database partitions 10,20,30,40,50,60, using a target ratio of 1:2:1:2:1:2. Note that in this example, only partitions 10, 20 and 30 actually exist in the database partition group

```
CALL SYSPROC.ANALYZE_LOG_SPACE('IBMDEFAULTGROUP', 'TEST',
    'EMP', 2, ' ', 'A', '40,50,60', '10,20,30,40,50,60',
    '1,2,1,2,1,2')
```

Analyze the effect of dropping a database partition without applying the changes. In the following case, the hypothesis is dropping database partition 30 from the database partition group, and redistributing the data in database partitions 10 and 20 using a target ratio of 1 : 1. Note that in this example, all database partitions 10, 20 and 30 should exist in the database partition group

```
CALL SYSPROC.ANALYZE_LOG_SPACE('IBMDEFAULTGROUP', 'TEST',
    'EMP', 2, ' ', 'D', '30', '10,20', '1,1')
```

Usage notes

"-1" is used as an output value for parameters when their values cannot be obtained.

The redistribute stored procedures and functions work only in partitioned database environments, where a distribution key has been defined for each table.

Information returned

The ANALYZE_LOG_SPACE procedure returns a result set (an open cursor) of the log space analysis results, containing the following fields for each of the database partitions of the given database partition group.

Table 203. Information returned by the ANALYZE_LOG_SPACE procedure

| Column name | Column type | Description |
|-----------------|-------------|--|
| PARTITION_NUM | SMALLINT | The database partition number of the log space analysis. |
| TOTAL_LOG_SIZE | BIGINT | Total log space allocated in bytes, -1 indicates unlimited size. |
| AVAIL_LOG_SPACE | BIGINT | The amount of log space in bytes that is free and can be used by the redistribute process. |
| DATA_SKEW | BIGINT | The absolute value in bytes of the size of data which is deviated from the target level. |
| REQ_LOG_SPACE | BIGINT | The amount of space in bytes required to reach the wanted data distribution. |
| NUM_OF_STEPS | SMALLINT | The number of steps needed to reduce the data skew to zero. |
| MAX_STEP_SIZE | BIGINT | The maximum amount of data in bytes that can be moved at a time, without causing a log full error. |

GENERATE_DISTFILE procedure - Generate a data distribution file

The GENERATE_DISTFILE procedure generates a data distribution file for the given table and saves it under the given fileName.

Syntax

►►—GENERATE_DISTFILE—(—*inTbSchema*—,—*inTbName*—,—*fileName*—)——————►►

The schema is SYSPROC.

Procedure parameters

inTbSchema

An input argument of type VARCHAR (128) that specifies the table schema name.

inTbName

An input argument of type VARCHAR (128) that specifies the table name.

fileName

An input or output argument of type VARCHAR (255) that specifies data distribution file name. If the given file name is just a file name, the file will be saved in the tmp sub-directory under the instance directory, and the full file path name will be returned in the parameter.

Authorization

- EXECUTE privilege on the GENERATE_DISTFILE procedure.
- SELECT privilege on SYSCAT.TABLES, SYSCAT.COLUMNS, and the specified table.

In addition, the fenced user ID must be able to create files in the **tmp** sub-directory under the instance directory.

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

Generate a data distribution file to be used by the redistribute process.

```
CALL SYSPROC.GENERATE_DISTFILE('TEST', 'EMP',  
    '$HOME/sql1lib/function/SAMPLE.IBMDEFAULTGROUP_swrData.dst')
```

Usage notes

The redistribute stored procedures and functions work only in partitioned database environments, where a distribution key has been defined for each table.

GET_SWRD_SETTINGS procedure - Retrieve redistribute information

The GET_SWRD_SETTINGS procedure reads the existing redistribute registry records for the given database partition group.

Syntax

```
►► GET_SWRD_SETTINGS (—dbpgName—, —matchingSpec—, —redistMethod—, —————►  
► pMapFile—, —distFile—, —stepSize—, —totalSteps—, —stageSize—, —————►  
► —nextStep—, —processState—, —pNumber—, —pWeight—) —————►◄
```

The schema is SYSPROC.

Procedure parameters

dbpgName

An input argument of type VARCHAR(128) that specifies the database partition group name against which the redistribute process is to run.

matchingSpec

An input argument of type SMALLINT that specifies the bitwise field identifier(s) from Table 204 on page 1004, indicating the target fields to be returned by the output parameters. Those output parameters that are not required can be set to null.

For example, if *matchingSpec* is set to 96, which is the integer value of (REDIST_STAGE_SIZE | REDIST_NEXT_STEP), the caller of this function only needs to provide *stageSize* and *nextStep* to receive the values, and the remainder of the output parameters can be null.

Table 204. Bitwise field identifiers

| Field Name | Hexadecimal value | Decimal value |
|---------------------------|-------------------|---------------|
| REDIST_METHOD | 0x0001<<0 | 1 |
| REDIST_PMAP_FILE | 0x0001<<1 | 2 |
| REDIST_DIST_FILE | 0x0001<<2 | 4 |
| REDIST_STEP_SIZE | 0x0001<<3 | 8 |
| REDIST_NUM_STEPS | 0x0001<<4 | 16 |
| REDIST_STAGE_SIZE | 0x0001<<5 | 32 |
| REDIST_NEXT_STEP | 0x0001<<6 | 64 |
| REDIST_PROCESS_STATE | 0x0001<<7 | 128 |
| REDIST_PWEIGHT_START_NODE | 0x0001<<8 | 256 |
| REDIST_PWEIGHT | 0x0001<<9 | 512 |

redistMethod

An output argument of type SMALLINT that specifies whether the redistribute is to run using the data distribution file or the target distribution map. There are two possible return values:

- 2: indicates that the redistribute process will work with a data distribution file as input.
- 3: indicates that the redistribute process will work with a target distribution map as input.

pMapFile

An output argument of type VARCHAR (255) that specifies the full path file name of the target distribution map on the database server.

distFile

An output argument of type VARCHAR (255) that specifies the full path file name of the data distribution file on the database server.

stepSize

An output argument of type BIGINT that specifies the maximum number of rows that can be moved before a commit must be called to prevent a log full situation. The number can be changed in each redistribution step.

totalSteps

An output argument of type SMALLINT that specifies the number of steps it takes to completely redistribute the given database partition group.

stageSize

An output argument of type SMALLINT that specifies the number of steps to be run consecutively.

nextStep

An output argument of type SMALLINT that specifies the index separating which steps have been completed, and what still needs to be run.

processState

An output argument of type SMALLINT that indicates whether or not the redistribute process will be stopped at the next check point. A check point is placed at beginning of each redistribute step. If this argument is set to 1, the step will not start; if the value is 0, the step will proceed.

pNumber

An output argument of type VARCHAR (6000) that might return a list of

comma-separated database partition numbers in a string format. These partition numbers can be either the database partitions that are currently used by the database partition group, or the ones to be added or dropped. The sequence and the count of these partition numbers correspond to the target partition weight returned by the *pWeight* variable.

pWeight

An output argument of type VARCHAR (6000) that might return a list of comma-separated target database partition weight numbers. The sequence and the count of these partition weights correspond to the partition numbers returned by the *pNumber* variable.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

Report the content of the step wise redistribution plan for the given database partition group.

```
CALL SYSPROC.GET_SWRD_SETTINGS
('IBMDEFAULTGROUP', 255, ?, ?, ?, ?, ?, ?, ?, ?, ?)
```

Usage note

The redistribute stored procedures and functions work only in partitioned database environments, where a distribution key has been defined for each table.

SET_SWRD_SETTINGS procedure - Create or change redistribute registry

The SET_SWRD_SETTINGS procedure creates or make changes to the redistribute registry.

If the registry does not exist, it creates it and add records into it. If the registry already exists, it uses *overwriteSpec* to identify which of the field values need to be overwritten. The *overwriteSpec* field enables this function to take NULL inputs for the fields that do not need to be updated.

Syntax

```
►►—SET_SWRD_SETTINGS—(—dbpgName—,—overwriteSpec—,—redistMethod—,——————►
►—pMapFile—,—distFile—,—stepSize—,—totalSteps—,—stageSize—,——————►
►—nextStep—,—processState—,—pNumber—,—pWeight—)——————►
```

The schema is SYSPROC.

Procedure parameters

dbpgName

An input argument of type VARCHAR(128) that specifies the database partition group name against which the redistribute process is to run.

overwriteSpec

Bitwise field identifier(s) from Table 205 indicating the target fields to be written or overwritten into the redistribute settings registry.

Table 205. Bitwise field identifiers

| Field Name | Hexadecimal value | Decimal value |
|---------------------------|-------------------|---------------|
| REDIST_METHOD | 0x0001<<0 | 1 |
| REDIST_PMAP_FILE | 0x0001<<1 | 2 |
| REDIST_DIST_FILE | 0x0001<<2 | 4 |
| REDIST_STEP_SIZE | 0x0001<<3 | 8 |
| REDIST_NUM_STEPS | 0x0001<<4 | 16 |
| REDIST_STAGE_SIZE | 0x0001<<5 | 32 |
| REDIST_NEXT_STEP | 0x0001<<6 | 64 |
| REDIST_PROCESS_STATE | 0x0001<<7 | 128 |
| REDIST_PWEIGHT_START_NODE | 0x0001<<8 | 256 |
| REDIST_PWEIGHT | 0x0001<<9 | 512 |

redistMethod

An input argument of type SMALLINT that specifies whether the redistribute is to run using the data distribution file or the target distribution map. The two valid input values are:

- 2: indicate that the redistribute process will work with a data distribution file as input.
- 3: indicate that the redistribute process will work with a target distribution map as input.

pMapFile

An input argument of type VARCHAR (255) that specifies the full path file name of the target distribution map on the database server.

distFile

An input argument of type VARCHAR (255) that specifies the full path file name of the data distribution file on the database server..

stepSize

An input argument of type BIGINT that specifies the maximum number of rows that can be moved before a commit must be called to prevent a log full situation. The number can be changed in each redistribution step. The value “-2” can be used for *stepSize* to indicate that the number is unlimited.

totalSteps

An input argument of type SMALLINT that specifies the number of steps it takes to completely redistribute the given database partition group. The value “-2” can be used *totalSteps* to indicate that the number is unlimited.

stageSize

An input argument of type SMALLINT that specifies the number of steps to be run consecutively.

nextStep

An input argument of type SMALLINT that specifies the index separating which steps have been completed, and what still needs to be run.

processState

An input argument of type SMALLINT that indicates whether or not the redistribute process will be stopped at the next check point. A check point is placed at beginning of each redistribute step. If this argument is set to 1, the step will not start; if the value is 0, the step will proceed.

pNumber

An input argument of type VARCHAR (6000) that can contain a list of comma-separated database partition numbers in a string format. These partition numbers can be either the database partitions that are currently used by the database partition group, or the ones to be added or dropped. The sequence and the count of these partition numbers correspond to the target partition weight returned by the *pWeight* variable. Each database partition number is between 0 and 999, and there are no spaces are allowed in the string.

pWeight

An input argument of type VARCHAR (6000) that can contain a comma-separated string of all the database partition weights the user has specified, corresponding to the database partition numbers in the *pNumber* string. Each database partition weight is a number between 0 and 32767, and no spaces are allowed in the string.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

Write a step wise redistribution plan into a registry. Setting *processState* to 1, might cause a currently running step wise redistribute stored procedure to complete the current step and stop, until this parameter is reset to 0, and the redistribute stored procedure is called again.

```
CALL SYSPROC.SET_SWRD_SETTINGS('IBMDEFAULTGROUP', 255, 0, ' ',
'$HOME/sql1lib/function/TEST.IBMDEFAULTGROUP_swrData.dst', 1000,
12, 2, 1, 0, '10,20,30', '50,50,50')
```

Usage notes

The redistribute stored procedures and functions work only in partitioned database environments, where a distribution key has been defined for each table.

STEPWISE_REDISTRIBUTE_DBPG procedure - Redistribute part of database partition group

The STEPWISE_REDISTRIBUTE_DBPG procedure redistributes part of the database partition group according to the input specified for the procedure, and the setting file created or updated by the SET_SWRD_SETTINGS procedure.

Syntax

```
►►STEPWISE_REDISTRIBUTE_DBPG—(—inDBPGroup—,—inStartingPoint—,——————►  
►—inNumSteps—)—————►►
```

The schema is SYSPROC.

Procedure parameters

inDBPGroup

An input argument of type VARCHAR (128) that specifies the name of the target database partition group.

inStartingPoint

An input argument of type SMALLINT that specifies the starting point to use. If the parameter is set to a positive integer and is not NULL, the STEPWISE_REDISTRIBUTE_DBPG procedure uses this value instead of using the *nextStep* value specified in the setting file. This is a useful option when you want to rerun the STEPWISE_REDISTRIBUTE_DBPG procedure from a particular step. If the parameter is set to NULL, the *nextStep* value is used.

inNumSteps

An input argument of type SMALLINT that specifies the number of steps to run. If the parameter is set to a positive integer and is not NULL, the STEPWISE_REDISTRIBUTE_DBPG procedure uses this value instead of using the *stageSize* value specified in the setting file. This is a useful option when you want to rerun the STEPWISE_REDISTRIBUTE_DBPG procedure with a different number of steps than what is specified in the settings. For example, if there are five steps in a scheduled stage, and the redistribution process failed at step 3, the STEPWISE_REDISTRIBUTE_DBPG procedure can be called to run the remaining three steps once the error condition has been corrected. If the parameter is set to NULL, the *stageSize* value is used. The value -2 can be used in this procedure to indicate that the number is unlimited.

Note: There is no parameter for specifying the equivalent of the **NOT ROLLFORWARD RECOVERABLE** option on the **REDISTRIBUTE DATABASE PARTITION GROUP** command. Logging is always performed for row data redistribution performed when the STEPWISE_REDISTRIBUTE_DBPG procedure is used.

Authorization

- EXECUTE privilege on the STEPWISE_REDISTRIBUTE_DBPG procedure
- SYSADM, SYSCtrl or DBADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

Redistribute the database partition group "IBMDEFAULTGROUP" according to the redistribution plan stored in the registry by the SET_SWRD_SETTINGS procedure. It is starting with step 3 and redistributes the data until 2 steps in the redistribution plan are completed.

```
CALL SYSPROC.STEPWISE_REDISTRIBUTE_DBPG('IBMDEFAULTGROUP', 3, 2)
```

For a full usage example of the stepwise redistribute procedures, refer to "Redistributing database partition groups using the STEPWISE_REDISTRIBUTE_DBPG procedure" in the *Partitioning and Clustering Guide*.

Usage notes

If the registry value for *processState* is updated to 1 using the SET_SWRD_SETTINGS procedure after the STEPWISE_REDISTRIBUTE_DBPG procedure execution is started, the process stops at the beginning to the next step and a warning message is returned.

As the SQL COMMIT statement is called by the redistribute process, running the redistribute process under a Type-2 connection is not supported.

Storage management tool routines

This grouping of routines and views can be used to interact with storage management objects.

CAPTURE_STORAGEMGMT_INFO procedure - Retrieve storage-related information for a given root object

The CAPTURE_STORAGEMGMT_INFO procedure attempts to collect the storage-related information for the given root object, as well as the storage objects defined within its scope.

All the storage objects are specified in the SYSTOOLS.STMG_OBJECT_TYPE table.

Table 206. STMG_OBJECT_TYPE table

| Column name | Data type | Nullable | Description |
|-------------|-----------|----------|---|
| OBJ_TYPE | INTEGER | N | Integer value corresponds to a type of storage object <ul style="list-style-type: none">• 0 - Database• 1 - Database Partition Group• 2 - Table Space• 3 - Table Space Container• 4 - Table• 5 - Index |

Table 206. *STMG_OBJECT_TYPE* table (continued)

| Column name | Data type | Nullable | Description |
|-------------|-----------|----------|---|
| TYPE_NAME | VARCHAR | N | Descriptive name of the storage object type <ul style="list-style-type: none"> • STMG_DATABASE • STMG_DBPGROUP • STMG_TABLESPACE • STMG_CONTAINER • STMG_TABLE • STMG_INDEX |

Authorization

- EXECUTE privilege on the CAPTURE_STORAGEMGMT_INFO procedure.
- EXECUTE privilege on the SYSPROC.DB_PARTITIONS, SYSPROC.MON_GET_TABLESPACE, SYSPROC.MON_GET_CONTAINER, and SYSPROC.WLM_SET_CONN_ENV routines.
- SELECT privilege on SYSCAT.TABLES, SYSCAT.TABLESPACES, SYSCAT.NODEGROUPDEF, SYSCAT.DATABASEPARTITIONS, SYSCAT.DATAPARTITIONEXPRESSION, SYSCAT.INDEXES, and SYSCAT.COLUMNS.

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Syntax

```
►►—CAPTURE_STORAGEMGMT_INFO—(—in_rootType—,—in_rootSchema—,——————►
►—in_rootName—)——————►◄◄
```

The schema is SYSPROC.

Procedure parameters

in_rootType

An input argument of type SMALLINT. The valid option types are:

- 0 - Database
- 1 - Database Partition Group
- 2 - Table Space
- 4 - Table
- 5 - Index

The input argument cannot be null. If a null value is specified, an SQL0443 error with SQLSTATE 38553, and token DBA7617 is returned.

in_rootSchema

An input argument of type VARCHAR (128) that specifies the schema name of the storage snapshot root object. A NULL value can be specified if the *in_rootType* is a database, a database partition group, or a table space.

in_rootName

An input argument of type VARCHAR (128) that specifies the name of the root object. The input argument cannot be null. If a null value is specified, an SQL0443 error with SQLSTATE 38553, and token DBA7617 is returned.

Usage notes

1. The following stored procedure should be used to create storage management tables:

`create_storagemgmt_tables(TABLESPACE_NAME)` where 'TABLESPACE' is the name of the table space, on which storage management tables would be created.

(In case of a problem with the existing storage management tables, it can be dropped using the following stored procedure and can be re-created using the preceding stored procedure.

`drop_storagemgmt_tables(0 or 1)` where '0' indicates 'stop' and '1' indicates 'proceed' on encountering an error.)

2. The following command should be used to run statistics for the storage object for which details have to be obtained:

`RUNSTATS ON TABLE (TABLESCHEMA.TABLENAME) ON KEY COLUMNS AND INDEXES ALL`

3. The following command should be used to populate the storage management tables:

Run 'capture_storagemgmt_info()' stored procedure to populate the storage management tables. Sometimes it might be necessary to run the CAPTURE_STORAGEMGMT_INFO procedure twice. The first time you run it, use the CAPTURE_STORAGEMGMT_INFO procedure to populate the storage tables with table space details. For example:

```
db2 "call capture_storagemgmt_info(0,<SCHEMA_NAME>,<DATABASE_NAME>)"
```

The second time, use the CAPTURE_STORAGEMGMT_INFO procedure to add details about the storage of the actual object to the storage table. For example, the following example adds details for an object of type index (the *in_rootType* argument is set to 5):

```
db2 "call  
capture_storagemgmt_info(5,<SCHEMA_NAME>,<SCHEMA_NAME.INDEX_NAME>)"
```

4. Run the select query on the required storage management table to see the details of the storage object, for example: as follows in case of INDEX object :

```
db2 "SELECT * FROM SYSTOOLS.STMG_INDEX"
```

CREATE_STORAGEMGMT_TABLES procedure - Create storage management tables

The CREATE_STORAGEMGMT_TABLES procedure creates all storage management tables under a fixed "DB2TOOLS" schema, in the table space specified by input.

Syntax

►►—CREATE_STORAGEMGMT_TABLES—(—*in_tbspace*—)—————►►

The schema is SYSPROC.

Procedure parameters

in_tbspace

An input argument of type VARCHAR(128) that specifies the table space name. The input argument cannot be null. If a null value is specified, an SQL0443 error with SQLSTATE 38553, and token DBA7617 is returned.

Authorization

EXECUTE privilege on the CREATE_STORAGEMGMT_TABLES procedure.

You must also have CREATETAB privilege on the database and USE privilege on the table space, and one of:

- IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name DB2TOOLS does not exist.
- CREATEIN privilege on the schema, if the schema name of the table exists.
- DBADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Usage notes

The following tables are created in the DB2TOOLS schema:

- STMG_CONTAINER
- STMG_CURR_THRESHOLD
- STMG_DATABASE
- STMG_DBPARTITION
- STMG_DBPGROUP
- STMG_HIST_THRESHOLD
- STMG_INDEX
- STMG_OBJECT
- STMG_OBJECT_TYPE
- STMG_ROOT_OBJECT
- STMG_TABLE
- STMG_TABLESPACE
- STMG_TBPARTITION
- STMG_THRESHOLD_REGISTRY

DROP_STORAGEMGMT_TABLES procedure - Drop all storage management tables

The DROP_STORAGEMGMT_TABLES procedure attempts to drop all storage management tables.

Syntax

►►—DROP_STORAGEMGMT_TABLES—(—dropSpec—)————►►

The schema is SYSPROC.

Procedure parameters

dropSpec

An input argument of type SMALLINT. When *dropSpec* is set to 0, the process stops when any error is encountered; when *dropSpec* is set to 1, the process continues, ignoring any error it encounters. The input argument cannot be null. If a null value is specified, an SQL0443 error with SQLSTATE 38553, and token DBA7617 is returned.

Authorization

EXECUTE privilege on the DROP_STORAGEMGMT_TABLES procedure.

The user ID that establishes the database connection must either be the definer of the storage management tables as recorded in the DEFINER column of SYSCAT.TABLES, or have at least one of the following privileges:

- DBADM authority
- DROPIN privilege on the schema for these tables
- CONTROL privilege on these tables

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Text search routines

This grouping of routines and views can be used to manage the Text Search feature and the text search indexes it uses.

SYSTS_ADMIN_CMD procedure - Run text search administration commands

The **SYSTS_ADMIN_CMD** procedure is used by applications to run text search administrative commands using the SQL CALL statement.

Authorization

EXECUTE privilege on the SYSTS_ADMIN_CMD procedure and the required authorization as listed for the requested operation.

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Syntax

►►SYSTS_ADMIN_CMD(—*command-string*—,—*message-locale*—,—*message*—)◀◀

The schema is SYSPROC.

Procedure parameters

command-string

An input argument of type VARCHAR (32 K) that specifies a single Text Search index administration command to be executed. The command syntax is

the same as the Text Search command except for the connection options, which are not supported through this procedure. Commands that you issue through this procedure use the current connection.

The procedure supports the following Text Search commands:

- **ALTER INDEX**
- **CLEAR COMMAND LOCKS**
- **CLEAR EVENTS**
- **CREATE INDEX**
- **DISABLE DATABASE**
- **DROP INDEX**
- **ENABLE DATABASE**
- **RESET PENDING**
- **UPDATE INDEX**

message-locale

An input argument of type VARCHAR(33) that specifies the language for any error message text that is returned. If the argument is null or an empty string or the message files for the specified locale are not available on the server, 'en_US' is used.

message

An output argument of type VARCHAR(32K) that specifies a warning or informational message for an operation that is considered successful.

Example

The following example updates the MYTEXTINDEX text search index in schema DB2TS and returns any error messages in English:

```
CALL SYSPROC.SYSTS_ADMIN_CMD  
('UPDATE INDEX DB2TS.MYTEXTINDEX FOR TEXT','en_US', ?);
```

Sample output is as follows:

Value of output parameters

Parameter Name : MESSAGE

Parameter Value : CIE00001 Operation completed successfully.

Return Status = 0

Usage notes

If the command does not run successfully, SQLCODE -20427 and SQLSTATE 38H14 are returned with the text-search-specific error message. For example, if index MYTEXTINDEX exists and the following statement is issued:

```
CALL SYSPROC.SYSTS_ADMIN_CMD ('CREATE INDEX MYTEXTINDEX FOR TEXT  
ON DB2TS.TEXTBOOKS (STORY)', 'en_US', ?)
```

Index creation fails with the following error message:

```
SQL20427N An error occurred during a text search administration  
procedure or command. The error message is "CIE00201 Text search  
index "DB2TS "."MYTEXTINDEX" already exists. ". SQLSTATE=38H14
```

If an SQLCODE is returned by the procedure, the message might be truncated. Full message information is in the **db2diag** log files.

SYSTS_ALTER procedure - Change the update characteristics of an index

This procedure is used to change the update characteristics of an index.

The procedure issues an **ALTER INDEX** text search administration command on the database server.

Authorization

The privileges that are held by the authorization ID of the statement must include the SYSTS_MGR role and at least one of the following authorities:

- DBADM authority
- ALTERIN privilege on base schema
- CONTROL or ALTER privilege on the base table on which the text search index is defined

To change an existing schedule, the authorization ID must be the same as the index creator, or must have DBADM authority.

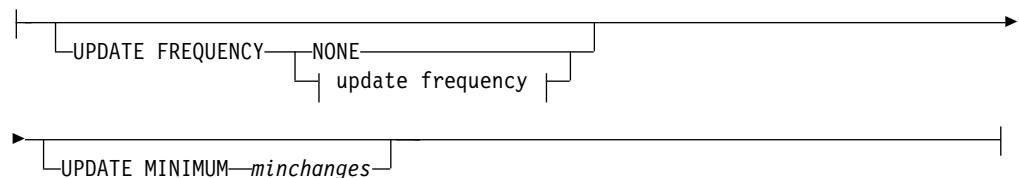
Default PUBLIC privilege

None

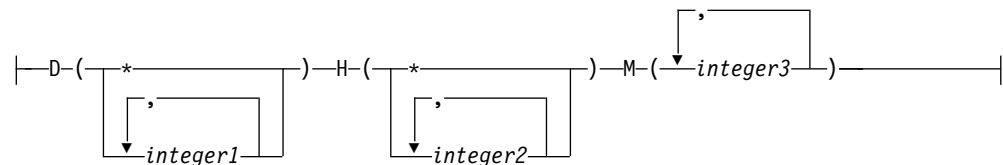
Syntax

```
►►SYSTS_ALTER(—index_schema—,—index_name—,— | update characteristics | ►►
► | options | —,—message_locale—,—message—) ►►
```

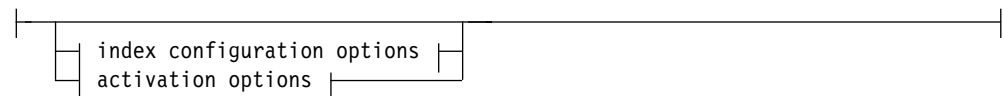
update characteristics:



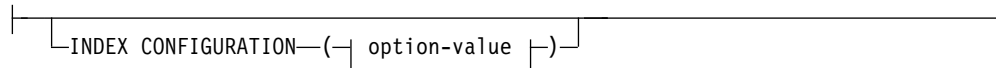
update frequency:



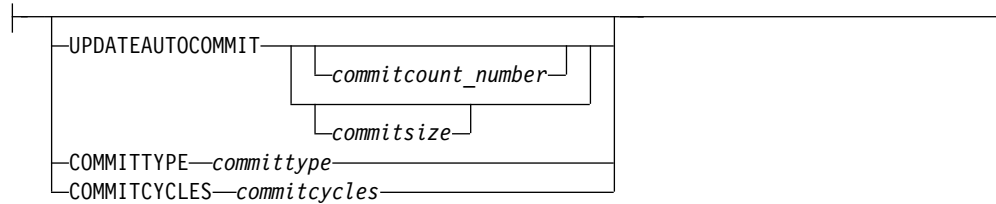
options:



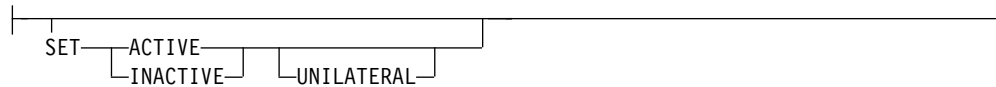
index configuration options:



option-value:



activation options:



The schema is SYSPROC.

Procedure parameters

index_schema

An input argument of type VARCHAR(128) that specifies the schema of the text search index. The *index_schema* must follow the naming restriction for database schema names. If the argument is null or an empty string, the value of the current schema is used. The *index_schema* is case-sensitive.

index_name

An input argument of type VARCHAR(128) that specifies the name of the index. Together with *index_schema*, it uniquely identifies a text search index in a database. The *index_name* is case-sensitive.

update characteristics

An input argument of type VARCHAR(32K) that specifies the alter options. The following alter options are allowed:

UPDATE FREQUENCY

Specifies the frequency with which index updates are made. The indexes are updated if the number of changes is at least the value that is set for **UPDATE MINIMUM**. The update frequency **NONE** indicates that no further index updates are made. This can be useful for a text column in a table with data that does not change. It is also useful if you intend to manually update the index by using the **UPDATE INDEX** command. You can do automatic updates if the DB2_ATS_ENABLE registry variable is set and you issue the **START FOR TEXT** command.

The default frequency value is taken from the view SYSIBMTS.TSDEFAULTS, where DEFAULTNAME='UPDATEFREQUENCY'.

NONE

No automatic updates are applied to the text index. You must start any further index updates manually.

- D** The days of the week when the index is updated.
- * Every day of the week.
- integer1***
Specific days of the week, from Sunday to Saturday: 0 - 6
- H** The hours of the specified days when the index is updated.
- * Every hour of the day.
- integer2***
Specific hours of the day, from midnight to 11 pm: 0 - 23
- M** The minutes of the specified hours when the index is updated.
- integer3***

If you do not specify the **UPDATE FREQUENCY** option, the frequency settings are unchanged.

UPDATE MINIMUM *minchanges*

Specifies the minimum number of changes to text documents that must occur before the index is incrementally updated. Multiple changes to the same text document are treated as separate changes. If you do not specify the **UPDATE MINIMUM** option, the setting is unchanged.

INDEX CONFIGURATION (*option-value*)

This is an optional input argument of type VARCHAR(32K) that allows altering text index configuration settings. The following options are supported:

Table 207. Specifications for option-value

| Option | Value | Data type | Description |
|---------------------|-------------------|-----------|---|
| SERIALUPDATE | <i>updatemode</i> | Integer | Specifies whether the update processing for a partitioned text search index must be run in parallel or in serial mode. In parallel mode the execution is distributed to the database partitions and issues independently on each node. In serial mode the execution is run without distribution and stops when a failure is encountered. Serial mode execution usually takes longer but requires less resources. <ul style="list-style-type: none"> • 0 = parallel mode • 1 = serial mode |

Table 207. Specifications for option-value (continued)

| Option | Value | Data type | Description |
|-------------------------|-------------------|-----------|--|
| UPDATEAUTOCOMMIT | <i>commitsize</i> | String | <p>Specifies the number of rows or number of hours after which a commit is run to automatically preserve the previous work for either initial or incremental updates.</p> <p>If you specify the number of rows:</p> <ul style="list-style-type: none"> After the number of documents that are updated reaches the COMMITCOUNT number, the server applies a commit. COMMITCOUNT counts the number of documents that are updated by using the primary key, not the number of staging table entries. <p>If you specify the number of hours:</p> <ul style="list-style-type: none"> The text index is committed after the specified number of hours is reached. The maximum number of hours is 24. <p>For initial updates, the index update processes batches of documents from the base table. After the <i>commitsize</i> value is reached, update processing completes a COMMIT operation and the last processed key is saved in the staging table with operational identifier '4'. This key is used to restart update processing either after a failure or after the number of specified <i>commitcycles</i> are completed. If a <i>commitcycles</i> is specified, the update mode is modified to incremental to initiate capturing changes by using the LOGTYPE BASIC option to create triggers on the text table. However, until the initial update is complete, log entries that are generated by documents that have not been processed in a previous cycle are removed from the staging table.</p> <p>Using the UPDATEAUTOCOMMIT option for an initial text index update leads to a significant increase of execution time.</p> <p>For incremental updates, log entries that are processed are removed correspondingly from the staging table with each interim commit.</p> |
| COMMITTYPE | <i>committype</i> | String | <p>Specifies rows or hours for the UPDATEAUTOCOMMIT index configuration option. The default is rows.</p> |

Table 207. Specifications for option-value (continued)

| Option | Value | Data type | Description |
|---------------------|---------------------|-----------|--|
| COMMITCYCLES | <i>commitcycles</i> | Integer | <p>Specifies the number of commit cycles. The default is 0 for unlimited cycles.</p> <p>If you do not explicitly specify cycles, the update operation uses as many cycles as required based on the batch size that you specified with the UPDATEAUTOCOMMIT option to finish the update processing.</p> <p>You can use this option with the UPDATEAUTOCOMMIT setting with a <i>committtype</i>.</p> |

activation options

Specifies an input argument of type integer that sets the status of a text index.

ACTIVE

Sets the text index status to active.

INACTIVE

Sets the text index status to inactive.

UNILATERAL

Specifies a unilateral change that affects the status of Text Search indexes. If you specify this argument, only the status of a Text Search index is changed to active or inactive. Without the UNILATERAL argument, the activation status of the Text Search and Net Search Extender indexes is jointly switched so that only one of the text indexes is active.

message_locale

An input argument of type VARCHAR(33) that specifies the locale to be used for any error message returned. If the argument is null or an empty string, or the message files for the specified locale are not available on the server, 'en_US' is used.

message

An output argument of type VARCHAR(32K) that specifies a warning or informational message for a successfully completed operation.

Examples

Example 1: In the following example, the update characteristics of a text search index are being altered. This index was originally created with *index_schema* 'db2ts' and *index_name* 'myTextIndex'. By using 'UPDATE FREQUENCY NONE', the intention is to make no further updates to the text search index as possibly no changes are expected for the associated table column. Any error messages are requested to be returned in English. When the procedure is complete, a success message is returned to the caller.

```
CALL SYSPROC.SYSTS_ALTER('db2ts', 'myTextIndex',
'UPDATE FREQUENCY NONE', 'en_US', ?)
```

The following example is of output from this query.

```

Value of output parameters
-----
Parameter Name   : MESSAGE
Parameter Value  : Operation completed successfully.

Return Status = 0

```

Example 2: In the following example, the SYSTS_ALTER stored procedure is called to alter the update-characteristics for a text search index with *index_schema* 'db2ts' and *index_name* 'myTextIndex2'. The intention is to ensure that updates to the index occur every hour on the hour. However, in this example the index does not exist and results in an error.

```

CALL SYSPROC.SYSTS_ALTER('db2ts', 'myTextIndex2',
'update frequency D(*) H(*) M(0)', 'en_US', ?)

```

The following example is of output from this query.

```

SQL20427N An error occurred during a text search administration
procedure or command. The error message is "CIE00316 Text search
index "db2ts"."myTextIndex2" does not exist. ". SQLSTATE 38H14

```

Usage notes

- Text search administration procedures use an existing connection to the database. You should commit all transaction changes before issuing a text search administration procedure to avoid any unexpected impact from a commit or rollback. One way to achieve this is to turn on AUTOCOMMIT.
- You cannot run multiple procedures or commands concurrently on a text search index if they might conflict. Some of the conflicting procedures and commands are:
 - SYSTS_ALTER procedure or ALTER INDEX db2ts command
 - SYSTS_CLEAR_EVENTS procedure or CLEAR EVENTS FOR INDEX db2ts command
 - SYSTS_DISABLE procedure or DISABLE DATABASE FOR TEXT db2ts command
 - SYSTS_DROP procedure or DROP INDEX db2ts command
 - SYSTS_UPDATE procedure or UPDATE INDEX db2ts command
 - SYSTS_CONFIGURE procedure

If there is a conflict, the procedure returns an SQLCODE -20426 and SQLSTATE 38H13.

- When you run this procedure to change the frequency, a schedule task is created, updated or dropped for the text index.
- The result of activating indexes depends on the original index status. The following table describes the results.

Table 208. Status changes without invalid index:

| Initial Text Search
or Net Search
Extender Status | Request
Active | Request Active
Unilateral | Request
Inactive | Request Inactive
Unilateral |
|---|----------------------|------------------------------|----------------------|--------------------------------|
| Active / Inactive | No change | No change | Inactive /
Active | Inactive /
Inactive |
| Inactive / Active | Active /
Inactive | Error | No change | No change |
| Inactive / Inactive | Active /
Inactive | Active / Inactive | Inactive /
Active | No change |

SQL20427N and CIE0379E error messages are returned for active index conflicts.

You can specify the **UPDATEAUTOCOMMIT** index configuration option without type and cycles for compatibility with an earlier version. This index configuration option is associated by default with the **COMMITTYPE** rows option and unrestricted cycles.

The **UPDATEAUTOCOMMIT**, **COMMITTYPE**, and **COMMITSIZE** index configuration options are also specified for an UPDATE INDEX operation to override the configured values. Values that you submit for a specific update operation are applied only once and not persisted.

SYSTS_CLEANUP procedure - Remove invalid text search indexes

This procedure removes invalid text search indexes and their associated collections from the database.

An index can become invalid when database operations are executed that affect all of the table content, for example, truncate.

The procedure serves as an alternative to the **db2ts CLEANUP FOR TEXT** command for the database scope. To remove obsolete collections on the instance level, the command must be used.

Authorization

The privileges held by the authorization ID of the statement must include the SYSTS_ADM role with DBADM and DATAACCESS authority.

Default PUBLIC privilege

None

Syntax

►►SYSTS_CLEANUP(—*options*—,—*message_locale*—,—*message*—)—————►◄

The schema is SYSPROC.

Procedure parameters

options

An input argument of type VARCHAR(32K). Only a NULL or an empty parameter is accepted. Reserved for internal IBM use.

message_locale

An input argument of type VARCHAR(33) that specifies the locale to be used for any error message returned. If the argument is null or an empty string, or the message files for the specified locale are not available on the server, 'en_US' is used.

message

An output argument of type VARCHAR(32K) that specifies a warning or informational message for a successfully completed operation.

Example

Example 1: In the following example, all invalid text search indexes in the current database are dropped. The message language is set to English, and when the procedure succeeds, the output parameter message indicative of the successful operation is returned to the caller.

```
CALL SYSPROC.SYSTS_CLEANUP('', 'en_US', ?)
```

Value of output parameters

Parameter Name : MESSAGE

Parameter Value : Operation completed successfully.

Return Status = 0

Example 2: Output where an error is returned:

SQL0462W Command or routine "SYSTS_CLEANUP" (specific name "*N") has returned a warning SQLSTATE, with diagnostic text "CIE00212W 2 of 20 collections could not be deleted. Check db2diag.log for details.". SQLSTATE=01H14

SYSTS_CLEAR_COMMANDLOCKS procedure - Remove command locks for text search indexes

This procedure removes all command locks for a specific text search index or for all text search indexes in the database.

Authorization

The privileges held by the authorization ID of the statement used to clear locks on the index must include both of the following authorities:

- SYSTS_MGR role
- DBADM authority or CONTROL privilege on the base table on which the index is defined

The privileges held by the authorization ID of the statement used to clear locks on the database connection must include the SYSTS_ADM role.

Default PUBLIC privilege

None

A command lock is created at the beginning of a text search index command, and is destroyed when the command has completed. It prevents undesirable conflict between different commands. Use of this procedure is required in the rare case that locks remain in place due to an unexpected system behavior, and need to be cleaned up explicitly.

This procedure issues the **CLEAR COMMAND LOCKS** text search administration command on the database server.

Syntax

```
►►SYSTS_CLEAR_COMMANDLOCKS(—index_schema—,—index_name—,——————►  
►message_locale—,—message—)—————►◄
```

The schema is SYSPROC.

Procedure parameters

index_schema

An input argument of type VARCHAR(128) that specifies the schema of the text index. The *index_schema* must follow the naming restriction for database schema names. If the argument is null or an empty string, the value of CURRENT SCHEMA is used. The *index_schema* is case-sensitive.

index_name

An input argument of type VARCHAR(128) that specifies the name of the index. Together with *index_schema*, it uniquely identifies a text search index in a database. If the argument is null or an empty string, the procedure deletes command locks for all text search indexes in the database. The *index_name* is case-sensitive.

message_locale

An input argument of type VARCHAR(33) that specifies the locale to be used for any error message returned. If the argument is null or an empty string, or the message files for the specified locale are not available on the server, 'en_US' is used.

message

An output argument of type VARCHAR(32K) that specifies a warning or informational message for a successfully completed operation.

Examples

Example 1: In the following example, SYSTS_CLEAR_COMMANDLOCKS is issued for a text search index with *index_schema* 'db2ts' and *index_name* 'myTextIndex'. Error messages are requested to be returned in English. When the procedure succeeds, the output parameter message indicative of the successful operation is returned to the caller.

```
CALL SYSPROC.SYSTS_CLEAR_COMMANDLOCKS('db2ts', 'myTextIndex', 'en_US', ?)
```

The following is an example of output from this query.

```
Value of output parameters
-----
Parameter Name  : MESSAGE
Parameter Value : Operation completed successfully.

Return Status = 0
```

Example 2: In the following example, SYSTS_CLEAR_COMMANDLOCKS is called to clear the command locks for a text search index with *index_schema* 'db2ts' and *index_name* 'myTextIndex2'. This index does not exist and the procedure returns an error message.

```
CALL SYSPROC.SYSTS_CLEAR_COMMANDLOCKS('db2ts', 'myTextIndex2', 'en_US', ?)
```

The following is an example of output from this query.

```
SQL20427N An error occurred during a text search administration
procedure or command. The error message is "CIE00316 Text search
index "db2ts"."myTextIndex2" does not exist. ". SQLSTATE 38H14
```

Usage notes

- Text search administration procedures use an existing connection to the database. It is recommended to commit all transaction changes before executing

a text search administration procedure to avoid any unexpected impact from a commit or rollback in the procedure. One way to achieve this is to turn on AUTOCOMMIT.

- You would invoke this procedure because the process owning the command lock is dead. In this case, the command (represented by the lock) may not have completed, and the index may not be operational. You need to take appropriate action. For example, the process executing the DROP INDEX command dies suddenly. It has deleted some index data, but not all the catalog and collection information. The command lock is left intact. After clearing the DROP INDEX command lock, you may want to re-execute the SYSTS_DROP procedure. In another example, the process executing the UPDATE INDEX command is interrupted. It has processed some documents, but not all, and the command lock is still in place. After reviewing the text search index status and clearing the UPDATE INDEX command lock, you can re-execute the UPDATE INDEX command.
- When this procedure is run, the content of the Text Search view SYSIBMTS.TSLOCKS is updated.

SYSTS_CLEAR_EVENTS procedure - Delete indexing events from an index's event table

This procedure deletes indexing events from an index's event table used for administration.

The name of the event table can be found in the view SYSIBMTS.TSINDEXES in column EVENTVIEWNAME. Every index update operation that processes at least one document produces informational and, in some cases, error entries in the event table. For automatic updates, the event table has to be regularly inspected. Document specific errors must be corrected by changing the document content. After correcting the errors, the events can be cleared (and should be, in order not to consume too much space).

The procedure issues a **CLEAR EVENTS FOR INDEX** text search administration command on the database server.

Authorization

The privileges held by the authorization ID of the statement must include both of the following authorities:

- SYSTS_MGR role
- DBADM with DATAACCESS authority or CONTROL privilege on the table on which the index is defined

Default PUBLIC privilege

None

Syntax

```
►►—SYSTS_CLEAR_EVENTS—(—index_schema—,—index_name—,——————►
►—message_locale—,—message—)—————►◄
```

The schema is SYSPROC.

Procedure parameters

index_schema

An input argument of type VARCHAR(128) that specifies the schema of the text search index. The *index_schema* must follow the naming restriction for database schema names. If the argument is null or an empty string, the value of CURRENT SCHEMA is used. The *index_schema* is case-sensitive.

index_name

An input argument of type VARCHAR(128) that specifies the name of the index. Together with *index_schema*, it uniquely identifies a text search index in a database. The *index_name* is case-sensitive.

message_locale

An input argument of type VARCHAR(33) that specifies the locale to be used for any error message returned. If the argument is null or an empty string, or the message files for the specified locale are not available on the server, 'en_US' is used.

message

An output argument of type VARCHAR(32K) that specifies a warning or informational message for a successfully completed operation.

Examples

Example 1: In the following example, SYSTS_CLEAR_EVENTS is being called for a text search index that was created with *index_schema* 'db2ts' and *index_name* 'myTextIndex'. Any error messages are requested to be returned in English. When the procedure succeeds, the output parameter message indicative of the successful operation is returned to the caller.

```
CALL SYSPROC.SYSTS_CLEAR_EVENTS('db2ts', 'myTextIndex', 'en_US', ?)
```

The following is an example of output from this query.

```
Value of output parameters
-----
Parameter Name   : MESSAGE
Parameter Value  : Operation completed successfully.

Return Status = 0
```

Example 2: In the following example, SYSTS_CLEAR_EVENTS is called to clear the event table entries for a text search index with *index_schema* 'db2ts' and *index_name* 'myTextIndex2'. This index does not exist and results in an error.

```
CALL SYSPROC.SYSTS_CLEAR_EVENTS('db2ts', 'myTextIndex2', 'en_US', ?)
```

The following is an example of output from this query.

```
SQL20427N An error occurred during a text search administration
procedure or command. The error message is "CIE00316 Text search
index "db2ts"."myTextIndex2" does not exist. ". SQLSTATE 38H14
```

Usage notes

- Text search administration procedures use an existing connection to the database. The current transaction might be committed or rolled back depending on the completion of the procedures. As such, you might want to commit all transaction changes to avoid any unexpected impact from such a commit or rollback. One way to achieve this is to turn on AUTOCOMMIT.

- Multiple procedures or commands cannot be run concurrently on a text search index if they might conflict. Some of the conflicting procedures and commands are:
 - SYSTS_ALTER procedure or ALTER INDEX db2ts command
 - SYSTS_DISABLE procedure or DISABLE DATABASE FOR TEXT db2ts command
 - SYSTS_DROP procedure or DROP INDEX db2ts command
 - SYSTS_CONFIGURE procedure
 - SYSTS_UPDATE procedure or UPDATE INDEX db2ts command

If there is a conflict, the procedure returns an SQLCODE -20426 and SQLSTATE 38H13.

- When regular updates are scheduled (see UPDATE FREQUENCY options in SYSTS_CREATE or SYSTS_ALTER procedures), the event table should be checked regularly.
- To clean up the Text Search event table for a text search index, use the SYSTS_CLEAR_EVENTS procedure or **CLEAR EVENTS FOR INDEX** db2ts command after you have checked the reason for the event and removed the source of the error.
- Ensure that changes have been made to all rows referenced in the event table. By changing the rows in the user table, you ensure that when you run the SYSTS_UPDATE procedure or **UPDATE INDEX** db2ts command again, an attempt is made to index the erroneous documents again.
- When this command is issued, the event table is cleared.

SYSTS_CONFIGURE procedure - Configure current database for text search

The **SYSTS_CONFIGURE** procedure applies text search server connection information to the text search catalog.

Certain text search server properties have to be reflected in the text search administration tables (text search catalog). These properties are associated with a text index when the index is created. When parameters such as the token are updated periodically, the change has to be reflected in the database and the properties for all the indexes have to be updated as well.

This procedure is required initially for:

- Incomplete enablement
- Stand-alone text search server setups
- Partitioned databases
- and further on, following any updates to text search server connection information

For subsequent updates, ensure that no text search administrative operation is active and shut down the currently configured text search server.

During database enablement the SYSIBMTS.TSSERVER administrative view is updated with connection information for the integrated text search server. Review and update the text server information in the SYSIBMTS.TSSERVER view with the relevant text search server data and run the SYSTS_CONFIGURE procedure to apply the updated information. For multiple databases in the instance, configure each database with the information for the same text search server.

Generally the sequence of operations is as follows:

1. Configure a text search server. Integrated text search servers can be configured during installation or when a database instance is created. A stand-alone text search server is configured separately.
2. Enable a database for text search by using the **db2ts ENABLE** command, or the **SYSTS_ENABLE** or **SYSTS_ADMIN_CMD** procedures with the **ENABLE** option.
3. Determine parameters for the text search server as needed for the **SYSIBMTS.TSSERVERS** view.
4. Update the **SYSIBMTS.TSSERVERS** administrative view with the parameters for the text search server.
 - If the view is empty then use an **INSERT** statement. For example:

```
INSERT INTO SYSIBMTS.TSSERVERS (HOST, PORT, TOKEN, SERVERSTATUS)
VALUES ('localhost', 55000, '9kfsjg48=', 0);
```
 - If the view already contains a row then use a **SQL UPDATE** statement. For example:

```
UPDATE SYSIBMTS.TSSERVERS SET (HOST, PORT, TOKEN) =
('tsmach1.ibm.com', 55002, 'k3j4fjk9u=')
```
5. Execute the **SYSTS_CONFIGURE** procedure.

Authorization

The privileges held by the authorization ID of the statement must include the **SYSTS_ADM** role.

Default PUBLIC privilege

None

Syntax

►►—**SYSTS_CONFIGURE**—(—*options*—,—*message-locale*—,—*message*—)————►►

The schema is **SYSPROC**.

Procedure parameter

options

An input argument of type **VARCHAR(32K)** that specifies the options to be used. If no options are needed, the argument can be null or an empty string.

message-locale

An input argument of type **VARCHAR(33)** that specifies the locale to be used for any returned error message. If the argument is null, an empty string or the message files for the specified locale are not available on the server, **en_US** is used.

message

An output argument of type **VARCHAR(32K)** that specifies a warning or informational message for an operation that is considered successful.

Example

Example 1: Prepare a database for text search services and return any messages in English.

```
CALL SYSTS_ENABLE('', 'en_US', ?)"

INSERT INTO SYSIBMTS.TSSERVERS
(HOST,PORT,TOKEN,KEY,LOCALE,SERVERTYPE,SERVERSTATUS)
VALUES ('tsmach1.ibm.com', 55000, '9kfsjg48=', 'en_US', 0);

CALL SYSPROC.SYSTS_CONFIGURE('', 'en_US', ?)
```

An example of output from this query:

```
Value of output parameters
-----
Parameter Name : MESSAGE
Parameter Value : Operation completed successfully.

Return Status = 0
```

Usage notes

- A SYSIBMTS.TSSERVERS view is created when a database is enabled for text search. This view is updated with information about the integrated text search server during database enablement. If the text search server configuration cannot be obtained, the enable operation will end with an 'incomplete enablement' warning.
 - The SYSTS_CONFIGURE procedure must be issued anytime a row is inserted or updated into SYSIBMTS.TSSERVERS. Make sure that no text search administration operation is active and shut down the text search server before updating SYSIBMTS.TSSERVERS.
 - When updating SYSIBMTS.TSSERVERS in a database, all text search enabled databases should be updated with the same parameters. Only a single text search server is supported with a given database instance.
 - Running the **SYSTS_CONFIGURE** procedure for a database registers the use of the configured text search server for the instance. Not running the procedure does not result in a severe error but the response to some commands can be unexpected.
 - Multiple procedures or commands cannot be executed concurrently on a text search index if they might conflict. Some of the conflicting procedures and commands are:
 - SYSTS_ALTER procedure
 - SYSTS_DISABLE procedure
 - SYSTS_CONFIGURE procedure
 - SYSTS_UPDATE procedure
- If there is a conflict, the procedure returns an SQLCODE -20426 and SQLSTATE 38H13.
- Certain aspects relating to the text search installation and database instance configuration for text search have to be updated. They include:
 - An indication whether the search server utilized by the database instance is integrated (configured by the database system as part of the database instance), or if it is a separate stand-alone installation of the ECMTS server.
 - An indication if the text search setup is enabled for rich text support.

SYSTS_CREATE procedure - Create a text search index on a column

The SYSTS_CREATE procedure creates a text search index for a text column by issuing the Text Search **CREATE INDEX** command on the database server. After you create and update a text search index, you can search the column data by using text search functions.

The index does not contain any data until an index update operation occurs. An update operation occurs in one of the following ways:

- You use the **SYSTS_CREATE** stored procedure.
- You issue the Text Search **UPDATE INDEX** command.
- The Administrative Task Scheduler issues the **UPDATE INDEX** command according to the update frequency that you defined for the index.

Authorization

The authorization ID of the **SYSTS_CREATE** stored procedure must hold the SYSTS_MGR role and CREATETAB authority on the database and one of the following items:

- CONTROL privilege on the table on which the index will be defined
- INDEX privilege on the table on which the index will be defined and one of the following items:
 - IMPLICIT_SCHEMA authority on the database, if the implicit or explicit schema name of the index does not exist
 - CREATEIN privilege on the schema, if the schema name of the index exists
- DBADM authority

Default PUBLIC privilege

None

Syntax

```
►►SYSTS_CREATE(—index_schema—,—index_name—,—| text source |—,——————►
►| options |—,—message_locale—,—message—)—————►◄
```

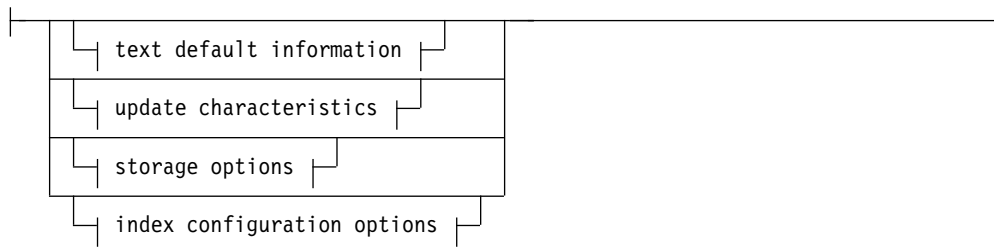
text source:

```
|—————| table_name (—| text column name |—)—,——————|
|——| table_schema |
```

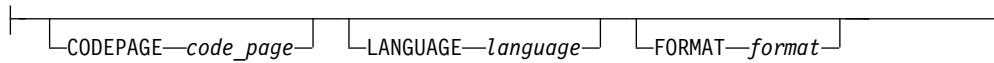
text column name:

```
|—————| column_name
|——| function_name (column_name) |
```

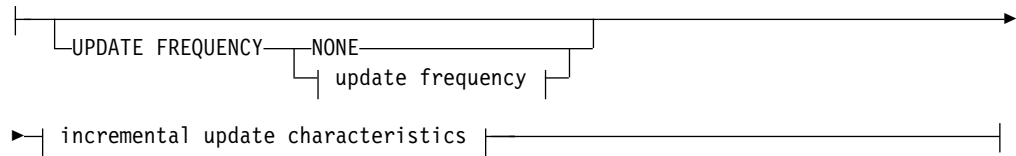
options:



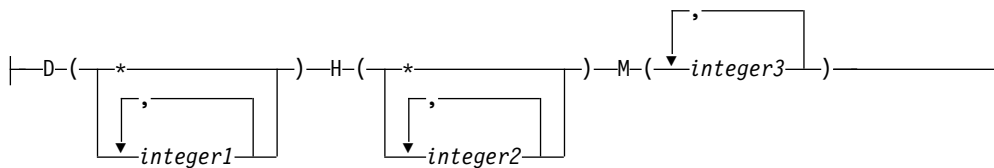
text default information:



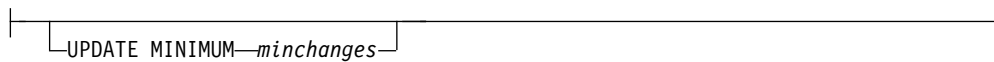
update characteristics:



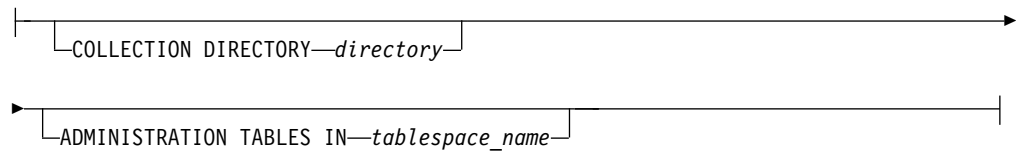
update frequency:



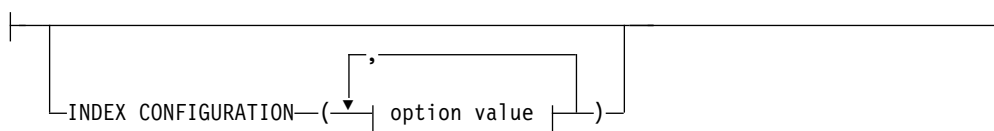
incremental update characteristics:



storage options:



index configuration options:



option value:

| | | | |
|-------------------------------|---|-------------------------------|-----------------------|
| COMMENT | — <i>text</i> — | | |
| UPDATEAUTOCOMMIT | <table><tr><td>—<i>commitcount_number</i>—</td></tr><tr><td>—<i>commitsize</i>—</td></tr></table> | — <i>commitcount_number</i> — | — <i>commitsize</i> — |
| — <i>commitcount_number</i> — | | | |
| — <i>commitsize</i> — | | | |
| COMMITTYPE | — <i>committype</i> — | | |
| COMMITCYCLES | — <i>commitcycles</i> — | | |
| INITIALMODE | — <i>initialmode</i> — | | |
| LOGTYPE | — <i>ltype</i> — | | |
| AUXLOG | — <i>auxlog_value</i> — | | |
| CJKSEGMENTATION | — <i>cjksegmentation_method</i> — | | |

The schema is SYSPROC.

Procedure parameters

index_schema

An input argument of type VARCHAR(128) that specifies the schema of the text search index. The *index_schema* argument must comply with the naming restrictions for database schema names. If the argument is null or an empty string, the default value of the schema is used. The *index_schema* argument is case-sensitive.

index_name

An input argument of type VARCHAR(128) that specifies the name of the index. Together with the *index_schema* argument, the *index_name* argument uniquely identifies a text search index in a database. The *index_name* argument is case-sensitive.

text source

An input argument of type VARCHAR(1024) that specifies the name of the column to index. The options are as follows:

table_schema

Specifies the schema of the table for which the text search index is to be created.

table_name

Specifies the name of the table that contains the text column. You can create a text search index on a nickname, but not on a federated table, materialized query table, or view. The *table_name* argument is case-sensitive.

text column name

Specifies the name of the column to index.

column_name

Specifies the name of the column to index. The column must be of one of the following data types: CHAR, VARCHAR, CLOB, DBCLOB, BLOB, GRAPHIC, VARGRAPHIC, or XML. If the data type of the column is not one of these data types, you can use a transformation function with the name *function_schema.function_name* to convert the column type to one of the valid types. (For details, see the *function_name(column_name)* option description.) Alternatively, you can specify a user-defined external function that accesses the text documents to be indexed.

You can create only a single text search index for a column. The *column_name* argument is case-sensitive.

function_name(column_name)

Specifies the schema-qualified name of an external scalar function that converts a column data type that is unsupported for text searching into a data type that is supported for text searching. This function must take only one parameter and return only one value. This option is case-sensitive.

options

An input argument of type VARCHAR(32K) that specifies the options to use. If you do not require any options, the parameter can be null or an empty string. The options are:

CODEPAGE *code_page*

Specifies the code page to use when indexing text documents. The default code page value is taken from the SYSIBMTS.TSDEFAULTS view, from the row with the DEFAULTNAME column value of CODEPAGE.

This option applies only to binary data types. That is, the column type or return type from a transformation function must be BLOB or BIT DATA.

LANGUAGE *language*

Specifies the language that Text Search uses for language-specific processing of a document during indexing. If you do not specify a locale, the database territory is used to determine the default setting for the LANGUAGE option. To have your documents scanned to determine the locale, specify *language* as AUTO .

FORMAT *format*

Specifies the format of the text documents in the column. The supported formats are TEXT, XML, HTML, and INSO. If the column data type is not XML, the default format is taken from the SYSIBMTS.TSDEFAULTS view, from the row with the DEFAULTNAME column value of FORMAT. If the column data type is XML, the default format is always XML. If you want to use the INSO format, you must install rich text support.

FORMAT *format*

Specifies the format of the text documents in the column. The supported formats are TEXT, XML, HTML, and INSO. If the column data type is not XML, the default format is taken from the SYSIBMTS.TSDEFAULTS view, from the row with the DEFAULTNAME column value of FORMAT. If the column data type is XML, the default format is always XML. If you want to use the INSO format, you must install rich text support.

UPDATE FREQUENCY

Specifies the frequency of index updates. The index is updated if the number of changes is at least the value of the UPDATE MINIMUM option. Automatic updates can be done only if you set the **DB2_ATS_ENABLE** registry variable and use the **START FOR TEXT** command.

The default format is taken from the SYSIBMTS.TSDEFAULTS view, from the row with the DEFAULTNAME column value of UPDATEFREQUENCY.

NONE

No further index updates are made. This value can be useful for a text column in a table with data that does not change. It is also useful if you intend to manually update the index by using the **UPDATE INDEX** command.

D The days of the week when the index is updated.

- * Every day of the week.

integer1

Specific days of the week, from Sunday - Saturday: 0 - 6.

H The hours of the specified days when the index is updated.

- * Every hour of the day.

integer2

Specific hours of the day, from midnight to 11 p.m.: 0 - 23.

M The minutes of the specified hours when the index is updated.

integer3

The top of the hour (0) , or 5-minute increments after the hour: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, or 55.

UPDATE MINIMUM *minchanges*

Specifies the minimum number of changes to text documents before the index is updated incrementally at the time that is specified by the UPDATE FREQUENCY option. Only positive integer values are allowed. The default value is taken from the SYSIBMTS.TSDEFAULTS view, from the row with the DEFAULTNAME column value of UPDATEMINIMUM. A small value increases consistency between the table column and the text search index but also causes additional load on the system. The **UPDATE INDEX** command ignores the value of this option unless you specify the USING UPDATE MINIMUM option for that command.

COLLECTION DIRECTORY *directory*

Specifies the directory in which the text search index collection is stored. You must specify the absolute path, where the maximum length of the absolute path name is 215 characters. The process owner of the instance service of the text search server must have read and write access to this directory.

The COLLECTION DIRECTORY option is supported only for an integrated text search server setup. For additional information about collection locations, review the usage notes.

ADMINISTRATION TABLES IN *table_space_name*

Specifies the name of an existing non-temporary table space for the administration tables that are created for the index. By default, the table space of the base table for which you are creating the index is used.

This argument is required when you are creating a text index for a partitioned table or for tables in partitioned databases. For partitioned databases, the table space must be in the same partition group as the table space of the base table. The text index administration tables are distributed in the same manner as the corresponding base table.

INDEX CONFIGURATION (*option_value*)

Specifies additional index-related options as option-value pairs. Options and values are as follows:

Table 209. Option-value pairs

| Option | Value | Data type | Description |
|------------------|-------------------|--------------------------------------|---|
| COMMENT | <i>text</i> | String value of fewer than 512 bytes | Adds a string comment to the REMARKS column in the Text Search SYSIBMTS.TSINDEXES catalog view. The comment is also used as the description of the collection. |
| UPDATEAUTOCOMMIT | <i>commitsize</i> | String | <p>Specifies the number of rows or number of hours after which a commit is run to preserve the previous work for either initial or incremental updates.</p> <p>If you specify the number of rows, after the number of updated documents reaches a COMMITCOUNT number, the server applies a commit. COMMITCOUNT counts the number of documents that are updated by using the primary key, not the number of staging table entries.</p> <p>If you specify the number of hours, the indexed data is committed after the specified number of hours is reached. The maximum number of hours is 24.</p> <p>For an initial update, the index update processes batches of documents from the base table. After the <i>commitsize</i> value is reached, update processing completes a commit operation, and the last processed key is saved in the staging table with the operational identifier 4. This key is used to restart update processing after a failure or after the completion of the specified number of <i>commitcycles</i>. If you specify a <i>commitcycles</i> value, the update mode is changed to incremental to initiate capturing changes by using the LOGTYPEBASIC option to create triggers on the text table. However, until the initial update is complete, log entries that were generated by documents that were not processed in a previous cycle are removed from the staging table.</p> <p>Using the UPDATEAUTOCOMMIT option for an initial text index update significantly increases execution time.</p> <p>For incremental updates, log entries that are processed are removed from the staging table with each interim commit.</p> |
| COMMITTYPE | <i>committype</i> | String | Specifies rows or hours for the UPDATEAUTOCOMMIT index configuration option. The default is rows. |

Table 209. Option-value pairs (continued)

| Option | Value | Data type | Description |
|--------------|---------------------|-----------|---|
| COMMITCYCLES | <i>commitcycles</i> | Integer | <p>Specifies the number of commit cycles. The default is 0, meaning unlimited cycles.</p> <p>If you do not specify the number of cycles, the update operation uses as many cycles as required to finish the update processing, based on the batch size that you specify for the UPDATEAUTOCOMMIT option.</p> <p>You can use the COMMITCYCLES option with the UPDATEAUTOCOMMIT option with a <i>committype</i> value.</p> |
| INITIALMODE | <i>initialmode</i> | String | <p>Specifies how the updates are processed. The possible values of the INITIALMODE index option are as follows:</p> <p>FIRST The first update is the default value of the INITIALMODE option.</p> <p>SKIP The update mode is immediately set to incremental, triggers are added for the LOGTYPEBASIC option, but no initial update is performed.</p> <p>NOW The update is started after the index is created as the final part of the CREATE INDEX command operation. This option is supported only for single-node setups.</p> |
| LOGTYPE | <i>ltype</i> | String | <p>Specifies whether triggers are added to populate the primary log table. The values are as follows:</p> <p>BASIC The primary staging table is created, and triggers are created on the text table to recognize any changes. This is the default value for text search indexes on base tables. This option is not supported for nicknames.</p> <p>CUSTOM The primary staging table is created, but no triggers are created on the text table. Identify changes for incremental updates, especially if you do not plan to use the ALLROWS option for updates. The CUSTOM option is supported for nicknames.</p> <p>Note: The default value of the LOGTYPE option is CUSTOM for text search indexes on nicknames.</p> |
| AUXLOG | <i>auxlog_value</i> | String | <p>Controls the creation of the additional log infrastructure to capture changes that are not recognized by a trigger. The default setting for range-partitioned tables is ON. You can change the default value in the default table by setting AuxLogNorm for non-range-partitioned tables and AuxLogPart for range-partitioned tables.</p> <p>For text search indexes on nicknames, only the OFF option can be enabled for the AUXLOG option to disable support for this index configuration option.</p> <p>You cannot change the auxiliary log infrastructure property for a text index after you create it.</p> |

Table 209. Option-value pairs (continued)

| Option | Value | Data type | Description |
|-----------------|-------------------------------|-----------|--|
| CJKSEGMENTATION | <i>cjksegmentation_method</i> | String | <p>Specifies the segmentation method that applies to documents that use the Chinese, Japanese, or Korean language (zh_CN, zh_TW, ja_JP, or ko_KR locale set), including such documents when automatic language detection is enabled (when you specify the LANGUAGE parameter with the AUTO option). Supported values are:</p> <ul style="list-style-type: none"> • MORPHOLOGICAL • NGRAM <p>If you do not specify a value, the value stored in the SYSIBMTS.TSDEFAULTS view is used. Specifically, the value in the DEFAULTVALUE column of the row whose DEFAULTNAME value is 'CJKSEGMENTATION'.</p> <p>The specified segmentation method is added to the SYSIBMTS.TSCONFIGURATION administrative view. You cannot change the method after creating the text index.</p> |

Important: You must enclose non-numeric values, such as comments, in single quotation marks. A single quotation mark character within a string value must be represented by two consecutive single quotation marks, as shown in the following example:

```
INDEX CONFIGURATION (COMMENT 'Index on User''s Guide column')
```

message_locale

An input argument of type VARCHAR(33) that specifies the locale to use for error messages. If the argument is null or an empty string or the message files for the specified locale are not available on the server, 'en_US' is used.

message

An output argument of type VARCHAR(32K) that specifies a warning or informational message for a successfully completed operation.

Examples

Example 1: In the following example, the SYSTS_CREATE procedure is called to create a text search index called myTextIndex. The **UPDATE MINIMUM 10** parameter specifies that 10 changes must be made to the text documents that are associated with the index before an incremental update of the index is performed. Any error messages are returned in English. When the underlying text search command runs successfully, the *message* output parameter is set to indicate the status of the command execution.

```
CALL SYSPROC.SYSTS_CREATE('db2ts', 'myTextIndex',
  'myUserSchema.myBaseTable (myTextColumn)', 'UPDATE MINIMUM 10',
  'en_US', ?)
```

Sample output is as follows:

```
Value of output parameters
-----
Parameter Name  : MESSAGE
Parameter Value : Operation completed successfully.
Return Status = 0
```

Example 2: In the following example, the SYSTS_CREATE procedure is called to create a text search index called myTextIndex2. No options are specified. In this example, the index exists, which results in an error message being returned to the caller.

```
CALL SYSPROC.SYSTS_CREATE('db2ts', 'myTextIndex2',  
    'myUserSchema.myBaseTable (myTextColumn)', '', 'en_US', ?)
```

Sample output is as follows:

```
SQL20427N An error occurred during a text search administration  
procedure or command. The error message is "CIE00201 Text search  
index "db2ts"."myTextIndex2" already exists. ".
```

Usage notes

- Text search administration procedures use an existing connection to the database. If a previous transaction was not committed before you perform a text search administration operation, a deadlock might occur because the same database objects might be affected and the operation waits for the previous transaction to be committed. This situation might occur, for example, when AUTOCOMMIT option is turned off and you create a table and then create a text index without an explicit commit between the two transactions.
- If you do not specify the **COLLECTION DIRECTORY** parameter, the collection is placed in a subdirectory that is named by using the system-generated index identifier in the path that you define by using the *defaultDataDirectory* parameter in the text server configuration. You can use the **configTool** utility to set this parameter. If you do not set the *defaultDataDirectory* parameter, the collection subdirectory is in the *configPath/config/collections* directory. You should specify a location other than the home directory or the database instance path for the *defaultDataDirectory* or **COLLECTION DIRECTORY** parameter. Make sure that the location has adequate storage space and is local, not NFS mounted.
- Configure a value for the **COLLECTION DIRECTORY** parameter that enables the use of a striped RAID device.

The following key-related rules apply:

- You must define a primary key for the table.
- The maximum number of primary key columns is two columns fewer than the maximum number of primary key columns in other environments.
- The maximum total length of all primary key columns for a table with Text Search indexes is 15 bytes fewer than the maximum total primary key length that is allowed in other environments. See the restrictions for the CREATE INDEX statement.

If you create an index with the **LANGUAGE** parameter set to the AUTO option, you can specify the CJKSEGMENTATION option.

You cannot issue certain procedures or commands concurrently for a text search index because they might result in an error, based on the timing of the conflicting operation:

- The SYSTS_DISABLE procedure or **db2ts DISABLE DATABASE FOR TEXT** command
- The SYSTS_CONFIGURE procedure

Successful execution of the **CREATE INDEX** command has the following effects:

- Text Search server data is updated. A collection with the name *instance_database-name_index_identifier_number* is created, as in the following example:

```
tigertail_MYTSDB_TS250517_0000
```

For partitioned databases, a collection is created for each partition. You can retrieve the collection names from the COLLECTIONNAME column in the SYSIBMTS.TSCOLLECTIONNAMES view.

- An index event table is created in the specified table space. Also, an index staging table is created in the specified table space with accurate database indexes. If you specified the AUXLOG ON option for the **INDEX CONFIGURATION** parameter, a second staging table is created to capture changes through integrity processing.
- Text Search catalog information is updated. However, the new text search index is not automatically populated. To populate the text search index, the **SYSTS_UPDATE** procedure or **UPDATE INDEX** command must be issued either manually or automatically (as a result of defining an update schedule for the index by using the **UPDATE FREQUENCY** parameter).
- If Text Search coexists with Net Search Extender and an active Net Search Extender index exists for the table column, the new Text Search index is set to inactive.
- If you create a text search index by setting the **LANGUAGE** parameter to AUTO and the CJKSEGMENTATION option to MORPHOLOGICAL, searches for valid strings on a morphological index return zero rows. To obtain the results, add the QUERYLANGUAGE option to the CONTAINS function, as shown in the following query:

```
select bookname from morphobooks
where contains (story, '△△','QUERYLANGUAGE=zh_CN') = 1
```

The AUXLOG option is supported for nicknames for data columns that support a materialized query table (MQT) with deferred refresh. It is not supported for views.

To create a text search index on a nickname, the nickname must be a non-relational flat file nickname. Non-relational XML nicknames are not supported.

For compatibility with an earlier version, you can specify the UPDATEAUTOCOMMIT index configuration option without type and cycles. This option is associated by default with the COMMITTYPE rows option and unrestricted cycles.

To override the configured values, you can specify the UPDATEAUTOCOMMIT, COMMITTYPE, and COMMITSIZE index configuration options for the **UPDATE INDEX** command. Values that you submit for a specific update operation are applied only once and are not persisted.

If you specify the INITIALMODE SKIP option, the text search index manager populates the index. Use this option to control the sequence in which data from the text table is initially processed.

The following rules apply to the LOGTYPE index configuration option:

- If you use the LOGTYPE CUSTOM setting, use the SYSIBMTS.TSSTAGING administrative view to insert log entries for new, changed, and deleted documents.

- To view the settings for an index, check the value of the LOGTYPE parameter in the SYSIBMTS.TSCONFIGURATION administrative view.
- To view the default log type that is applied to new text indexes, check the value of the LOGTYPE option in the SYSIBMTS.TSDEFAULTS administrative view.
- The LOGTYPE option is not valid with the ALLROWS option of the **CREATE INDEX** command because the ALLROWS option forces an initial update and no log tables are created.

SYSTS_DISABLE procedure - Disable current database for text search

The procedure disables Text Search for the current database.

Once the Text Search feature has been disabled, text search indexes and commands are no longer available for use with the database.

The procedure issues a **DISABLE DATABASE FOR TEXT** text search administration command on the database server.

Authorization

The privileges held by the authorization ID of the statement must include the following authorities:

- DBADM with DATAACCESS authority.
- SYSTS_ADM role

Default PUBLIC privilege

None

Syntax

►►SYSTS_DISABLE—(—*options*—,—*message_locale*—,—*message*—)—————►►

The schema is SYSPROC.

Procedure parameters

options

An input argument of type VARCHAR(128) that specifies the options to be used when disabling the database. The argument can be set to FORCE. When this value is specified, all indexes are dropped and the Text Search feature is disabled by force. No text search indexes are preserved and no error message or warning is returned. If the argument is null or an empty string, an attempt is made to disable the Text Search feature for the database.

message_locale

An input argument of type VARCHAR(33) that specifies the locale to be used for any error message returned. If the argument is null or an empty string, or the message files for the specified locale are not available on the server, 'en_US' is used.

message

An output argument of type VARCHAR(32K) that specifies a warning or informational message for a successfully completed operation.

Examples

Example 1: In the following example, Text Search is disabled for a database using the SYSTS_DISABLE procedure. The FORCE option is specified to ensure that the feature is disabled even if text search indexes still exist on tables in the database. Error messages are specified to be returned in English. The *message* output parameter is set to an informational message string.

```
CALL SYSPROC.SYSTS_DISABLE('FORCE', 'en_US', ?)
```

The following output is an example of sample output from this query.

```
Value of output parameters
-----
Parameter Name   : MESSAGE
Parameter Value  : Operation completed successfully.

Return Status = 0
```

Example 2: In the following example, Text Search is disabled for a database with existing text search indexes using the SYSTS_DISABLE procedure without specifying the FORCE option. This results in an error message to the caller. It is preferable to drop all existing text search indexes before disabling the Text Search feature or alternatively to specify the FORCE option for the *options* input parameter value.

```
CALL SYSPROC.SYSTS_DISABLE('', 'en_US', ?)
```

The following output is an example of sample output from this query.

```
SQL20427N An error occurred during a text search administration
procedure or command. The error message is "CIE00326 Text search
index active in specified or default database. ". SQLSTATE 38H14
```

Usage notes

- Text search administration procedures use an existing connection to the database. It is recommended to commit all transaction changes before executing a text search administration procedure to avoid any unexpected impact from a commit or rollback in the procedure. One way to achieve this is to turn on AUTOCOMMIT.
- Multiple procedures or commands cannot be executed concurrently on a text search index if they might conflict. Some of the conflicting procedures are:
 - SYSTS_ALTER procedure
 - SYSTS_CLEAR_EVENTS procedure
 - SYSTS_DISABLE procedure
 - SYSTS_CONFIGURE procedure
 - SYSTS_UPDATE procedure

If there is a conflict, the procedure returns an SQLCODE -20426 and SQLSTATE 38H13.

Note: The lock is set at the database level for the **SYSTS_DISABLE** procedure.

- When this procedure is run,
 - the Text Search catalog information is updated. The index log and event tables are dropped. Triggers on the user text table are deleted.
 - if the **FORCE** option is specified, all text index information is removed from the database and all associated collections are deleted. See the "db2ts DROP INDEX command" or "SYSTS_DROP procedure" for reference.

- This procedure does not influence the Net Search Extender enablement status of the database. It deletes the Text Search catalog tables and views that are created by the SYSTS_ENABLE procedure or the **ENABLE FOR TEXT** command.
- Before dropping a database that has text search index definitions, run this procedure and make sure that the text indexes and collections have been removed successfully.
- If some indexes could not be deleted using the **FORCE** option, the collection names are written to the **db2diag** log files.

Note: The user is discouraged from usage that results in orphaned collections, that is, collections that remain defined on the text search server but are not used by the database system. Here are some cases that may cause orphaned collections:

- When a DROP DATABASE CLP command is executed without running a DISABLE DATABASE FOR TEXT command
- When the SYSTS_DISABLE procedure is run or a **DISABLE DATABASE FOR TEXT** command is executed using the **FORCE** option.

SYSTS_DROP procedure - Drop text search index

The SYSTS_DROP procedure drops an existing text search index associated with any table column.

After this procedure runs successfully, text search queries cannot be run on the column for which the text search index was dropped.

The procedure issues a **DROP INDEX** text search administration command on the database server.

Authorization

The privileges held by the authorization ID of the statement must include the SYSTS_MGR role and at least one of the following privileges or authorities:

- CONTROL privilege on the table on which the index is defined
- DROPIN privilege on the schema on which the index is defined
- If the text search index has an existing schedule, the authorization ID must be the same as the index creator, or must have DBADM authority

Default PUBLIC privilege

None

Syntax

```
►►SYSTS_DROP(—index_schema—,—index_name—,—options—►►
►—message_locale—,—message—)►►
```

The schema is SYSPROC.

Procedure parameters

index_schema

An input argument of type VARCHAR(128) that specifies the schema of the text search index. The *index_schema* must follow the naming restriction for

database schema names. If the argument is null or an empty string, the value of CURRENT SCHEMA is used. The *index_schema* is case-sensitive.

index_name

An input argument of type VARCHAR(128) that specifies the name of the index. Together with *index_schema*, it uniquely identifies a text search index in a database. The *index_name* is case-sensitive.

options

An input argument of type VARCHAR(32000) that specifies the options to be used. If no options are needed, the argument can be null or an empty string.

message_locale

An input argument of type VARCHAR(33) that specifies the locale to be used for any error message returned. If the argument is null or an empty string, or the message files for the specified locale are not available on the server, 'en_US' is used.

message

An output argument of type VARCHAR(32K) that specifies a warning or informational message for a successfully completed operation.

Examples

Example 1: In the following example, the text search index that was created with *index_schema* 'db2ts' and *index_name* 'myTextIndex' is being dropped. Any error messages are requested to be returned in English. When the procedure succeeds, the output parameter message indicative of the successful operation is returned to the caller.

```
CALL SYSPROC.SYSTS_DROP('db2ts', 'myTextIndex', '', 'en_US', ?)
```

As in previous releases, the SYSTS_DROP procedure is supported without the **options** argument, for example :

```
CALL SYSPROC.SYSTS_DROP('db2ts', 'myTextIndex', 'en_US', ?)
```

The following is an example of output from this query.

```
Value of output parameters
-----
Parameter Name   : MESSAGE
Parameter Value  : Operation completed successfully.

Return Status = 0
```

Example 2: In the following example, SYSTS_DROP is called to drop a text search index with *index_schema* 'db2ts' and *index_name* 'myTextIndex2'. This index does not exist and results in an error.

```
CALL SYSPROC.SYSTS_DROP('db2ts', 'myTextIndex2', 'en_US', ?)
```

The following is an example of output from this query.

```
SQL20427N An error occurred during a text search administration
procedure or command. The error message is "CIE00316 Text search
index "db2ts"."myTextIndex2" does not exist. ". SQLSTATE 38H14
```

Usage notes

- Multiple procedures or commands cannot be executed concurrently on a text search index if they might conflict. Some of the conflicting procedures are:
 - SYSTS_ALTER procedure
 - SYSTS_CLEAR_EVENTS procedure

- SYSTS_DISABLE procedure
- SYSTS_DROP procedure
- SYSTS_UPDATE procedure
- SYSTS_CONFIGURE procedure

A STOP FOR TEXT command that runs in parallel with the DROP operation will not cause a conflicting command message, instead, if the text search server is shut down before DROP has removed the collection, an error will be returned that the text search server is not available.

- After a text search index is dropped, text search is no longer possible on the corresponding text column. If you plan to create a new text search on the same text column, you must first disconnect from the database and then reconnect before creating the new text search index.
- The db2ts DROP INDEX command makes the following changes to the database:
 - Updates the Text Search catalog information.
 - Drops the index staging and event tables.
 - Deletes triggers on the user text table.
 - Destroys the collection associated with the Text Search index definition.

SYSTS_ENABLE procedure - Enable current database for text search

The SYSTS_ENABLE procedure enables Text Search for the current database.

This procedure must be issued successfully before text search indexes on columns within the database can be created.

This procedure issues the **ENABLE DATABASE FOR TEXT** text search administration command on the database server.

Authorization

The privileges held by the authorization ID of the procedure must include the SYSTS_ADM role and the DBADM authority.

Default PUBLIC privilege

None

Syntax

►►SYSTS_ENABLE—(—options—,—message_locale—,—message—)————►

options:

►►└┐ ADMINISTRATION TABLES IN tablespace-name └┐————►

The schema is SYSPROC.

Procedure parameters

options

An input argument of type VARCHAR(32K) that specifies the options to be used. If no options are needed, the argument can be null or an empty string. Supported values are:

ADMINISTRATION TABLES IN *table-space-name*

Specifies the name of an existing regular table space for administration tables created while enabling the database for Text Search. If this clause is not specified, SYSTOOLSPACE is used as the table space. Regardless of whether SYSTOOLSPACE or an explicitly specified table space name is used, the table space has to meet certain requirements. It has to be:

- A regular table space
- For partitioned databases it would be recommended to define a bufferpool and table space with 32KB page size

To use a case-sensitive tablespace-name use a delimited identifier by enclosing the name in double quotation marks. By default it is treated as an ordinary identifier and converted to uppercase.

message_locale

An input argument of type VARCHAR(33) that specifies the locale to be used for any error message returned. If the argument is null or an empty string, or the message files for the specified locale are not available on the server, 'en_US' is used.

message

An output argument of type VARCHAR(32K) that specifies a warning or informational message for a successfully completed operation.

Examples

Example 1: Enable the database for text search by creating administration tables in a table space with any output messages in English.

```
CALL SYSPROC.SYSTS_ENABLE ('ADMINISTRATION TABLES IN TSPACE', 'en_US', ?)
```

Enable the database for text search by creating administration tables in a table space with any output messages in French.

```
CALL SYSPROC.SYSTS_ENABLE ('ADMINISTRATION TABLES IN "tbs32k" ', 'fr_FR', ?)
```

As in previous releases, the SYSTS_ENABLE procedure is supported without the **options** argument, for example:

```
CALL SYSPROC.SYSTS_ENABLE ('en_US', ?)
```

An example of output for this query:

```
Value of output parameters
-----
Parameter Name  : MESSAGE
Parameter Value : Operation completed successfully.

Return Status = 0
```

Example 2: In the following example, SYSTS_ENABLE is called on a database that is already enabled for text search. This results in an error message to the caller.

```
CALL SYSPROC.SYSTS_ENABLE('en_US', ?)
```

An example of output for this query:

SQL20427N An error occurred during a text search administration procedure or command. The error message from the text search product is "CIE00322 Specified or default database already enabled for text. ". SQLSTATE 38H14

Usage notes

- Text search administration procedures use an existing connection to the database. It is recommended to commit all transaction changes before executing a text search administration procedure to avoid any unexpected impact from a commit or rollback in the procedure. One way to achieve this is to turn on AUTOCOMMIT.
- When this procedure is run, the following events occur:
 - This procedure creates database objects, such as text search administration catalog tables and views, in the schema SYSIBMTS.
 - The established database defaults for text search index are available in view SYSIBMTS.TSDEFAULTS.
 - When the command has successfully completed, the text search catalog tables and views are created and are available.
- When executing this procedure, if you do not have sufficient execution and file access privileges to retrieve the Text Search server configuration, the procedure will create the text search catalog with an 'Incomplete enablement' warning. In this case, the Text Search server connection information will need to be updated manually.

SYSTS_UPDATE procedure - Update the text search index

The **SYSTS_UPDATE** procedure updates the text search index to reflect the current contents of the text column with which the index is associated.

While the update is being done, a search is possible. Until completion of the update, the search operates on a partially updated index.

The procedure issues an **UPDATE INDEX** text search administration command on the database server.

Authorization

The privileges that are held by the authorization ID of the statement must include the **SYSTS_MGR** role and at least one of the following authorities:

- **DATAACCESS** authority
- **CONTROL** privilege on the table on which the text index is defined
- **INDEX** with **SELECT** privilege on the base table on which the text index is defined

In addition, for an initial update the authorization requirements apply as outlined in the **CREATE TRIGGER** statement.

Default PUBLIC privilege

None

Syntax

►►SYSTS_UPDATE(—*index_schema*—,—*index_name*—,——————►

►*—update_options—,—message_locale—,—message—*)◄

The schema is SYSPROC.

Procedure parameters

index_schema

An input argument of type VARCHAR(128) that specifies the schema of the text search index. The *index_schema* must follow the naming restriction for database schema names. If the argument is null or an empty string, the value of the current schema is used. The *index_schema* is case-sensitive.

index_name

An input argument of type VARCHAR(128) that specifies the name of the index. Together with *index_schema*, it uniquely identifies a text search index in a database. The *index_name* is case-sensitive.

update_options

An input argument of type VARCHAR(32K) that specifies update options. If no options are required the argument can be null or an empty string. The possible values are:

| UPDATE OPTIONS value | Description |
|-------------------------|--|
| USING UPDATE MINIMUM | This option enforces the use of the UPDATE MINIMUM value that is defined for the text search index and processes updates if the specified minimum number of changes occurred. |
| FOR DATA REDISTRIBUTION | This option specifies that a text search index in a partitioned database must be refreshed after data partitions are added or removed and a subsequent data redistribution operation must be completed. Search results might be inconsistent until the text search index is updated with the FOR DATA REDISTRIBUTION option. |
| ALLROWS | This option specifies that an initial update must be attempted unconditionally. |

| UPDATE OPTIONS value | Description |
|------------------------------------|--|
| UPDATEAUTOCOMMIT <i>commitsize</i> | <p>Specifies the number of rows or number of hours after which a commit is run to automatically preserve the previous work for either initial or incremental updates.</p> <p>If you specify the number of rows:</p> <ul style="list-style-type: none"> • After the number of documents that are updated reaches the COMMITCOUNT number, the server applies a commit. COMMITCOUNT counts the number of documents that are updated by using the primary key, not the number of staging table entries. <p>If you specify the number of hours:</p> <ul style="list-style-type: none"> • The text index is committed after the specified number of hours is reached. The maximum number of hours is 24. <p>For initial updates, the index update processes batches of documents from the base table. After the <i>commitsize</i> value is reached, update processing completes a COMMIT operation and the last processed key is saved in the staging table with operational identifier '4'. This key is used to restart update processing either after a failure or after the number of specified <i>commitcycles</i> are completed. If a <i>commitcycles</i> is specified, the update mode is modified to incremental to initiate capturing changes by using the LOGTYPE BASIC option to create triggers on the text table. However, until the initial update is complete, log entries that are generated by documents that have not been processed in a previous cycle are removed from the staging table.</p> <p>Using the UPDATEAUTOCOMMIT option for an initial text index update leads to a significant increase of execution time.</p> <p>For incremental updates, log entries that are processed are removed correspondingly from the staging table with each interim commit.</p> |
| COMMITTYPE | <p>Specifies rows or hours for the UPDATEAUTOCOMMIT index configuration option. The default is rows.</p> |

| UPDATE OPTIONS value | Description |
|----------------------|--|
| COMMITCYCLES | <p>Specifies the number of commit cycles. The default is 0 for unlimited cycles.</p> <p>If cycles are not explicitly specified, the update operation uses as many cycles as required based on the batch size that is specified with the UPDATEAUTOCOMMIT option to finish the update processing.</p> <p>You can use this option with the UPDATEAUTOCOMMIT setting with a <i>committtype</i>.</p> |

message_locale

An input argument of type VARCHAR(33) that specifies the locale to be used for any error message returned. If the argument is null or an empty string, or the message files for the specified locale are not available on the server, 'en_US' is used.

message

An output argument of type VARCHAR(32K) that specifies a warning or informational message for a successfully completed operation.

Examples

Example 1: In the following example, the text search index that was created with *index_schema* 'db2ts' and *index_name* 'myTextIndex' is being updated. A NULL value in the place of the *update_options* means that an update is unconditionally started when the stored procedure is called. Any error messages are requested to be returned in English. For successful procedures, the caller receives a success message.

```
CALL SYSPROC.SYSTS_UPDATE
('db2ts', 'myTextIndex', '', 'en_US', ?)
```

An example of output from this query:

```
Value of output parameters
-----
Parameter Name  : MESSAGE
Parameter Value : Operation completed successfully.
```

Example 2: Update a text index after an operation finishes on the partition group that is associated with the base table, and return any error messages in English.

```
CALL SYSPROC.SYSTS_UPDATE
('db2ts', 'myTextIndex2', 'FOR DATA REDISTRIBUTION', 'en_US', ?)
```

```
Value of output parameters
-----
Parameter Name  : MESSAGE
Parameter Value : Operation completed successfully.
```

```
Return Status = 0
```

Example 3: In the following example, SYSTS_UPDATE is called to update a text search index with *index_schema* 'db2ts' and *index_name* 'myTextIndex3'. In this example, the index does not exist and results in an error.

```
CALL SYSPROC.SYSTS_UPDATE('db2ts', 'myTextIndex3', 'USING UPDATE MINIMUM',
'en_US', ?)
```

SQL20427N An error occurred during a text search administration procedure or command. The error message is "CIE00316 Text search index "db2ts"."myTextIndex3" does not exist. ". SQLSTATE 38H14

Usage notes

- Text search administration procedures use an existing connection to the database. The current transaction might be committed or rolled back depending on the completion of the procedures. To avoid any unexpected impact from such a commit or rollback, you might want to commit all transaction changes. Turning on AUTOCOMMIT is one way to commit all transaction changes.
- Certain procedures or commands cannot be executed concurrently on a text search index because the timing of the conflicting operation might cause an error. Some of the conflicting procedures and commands are:
 - SYSTS_ALTER procedure or **db2ts ALTER INDEX** command
 - SYSTS_CLEAR_EVENTS procedure or **db2ts CLEAR EVENTS FOR INDEX** command
 - SYSTS_DISABLE procedure or **db2ts DISABLE DATABASE FOR TEXT** command
 - SYSTS_UPDATE procedure or **db2ts UPDATE INDEX** command

If there is a conflict, the procedure returns an SQLCODE -20426 and SQLSTATE 38H13.

- This procedure does not return until all index update processing is completed. The duration depends on the number of documents to be indexed and the number of documents already indexed. The collection name for the index can be retrieved from the SYSIBMTS.TSCOLLECTIONNAMES view COLLECTIONNAME column.
- When there are individual document errors, the documents must be corrected. The primary keys of the erroneous documents can be looked up in the event table for the index. By changing the corresponding rows in the user table, the next call to SYSTS_UPDATE reprocesses these documents.
- When the SYSTS_UPDATE procedure is run, the following events occur:
 - Rows are inserted into the event table, including parser error information. Information is deleted from the index staging table in case of incremental updates. Before the first update, the SYSTS_UPDATE procedure creates triggers on the user table.
 - The collection is updated:
 - New or changed documents are parsed and indexed.
 - Deleted documents are discarded from the index.
- If a synonym dictionary is associated with a text index, issuing the update with the ALLROWS or FOR DATA REDISTRIBUTION options removes the association by dropping and re-creating the collections for the text index. The synonym dictionary must be associated with the new text index collections.
- The **UPDATEAUTOCOMMIT** index configuration option can be specified without type and cycles for compatibility with an earlier version. It is associated by default with the **COMMITTYPE** rows option and unrestricted cycles.
- When **UPDATEAUTOCOMMIT**, **COMMITTYPE** or **COMMITSIZE** values are specified for the update operation, they override existing configured values only for the specific update and are not persisted. update.

SYSTS_UPGRADE_CATALOG procedure - Upgrade the text search catalog

This procedure upgrades the Text Search catalog, including the administrative tables and administrative views, to the latest product version.

This procedure creates new catalog tables and views which are used in the latest version of the product and also updates the existing catalog tables and views. It removes obsolete catalog tables and views.

Authorization

The privileges held by the authorization ID of the procedure must include the SYSTS_ADM role and the DBADM authority.

Default PUBLIC privilege

None

Syntax

►►SYSTS_UPGRADE_CATALOG(—*message_locale*—,—*message*—)————►►

The schema is SYSPROC.

Procedure parameters

message_locale

An input argument of type VARCHAR(33) that specifies the locale to be used for any error message returned. If the argument is null or an empty string, or the message files for the specified locale are not available on the server, 'en_US' is used.

message

An output argument of type VARCHAR(32K) that specifies a warning or informational message for a successfully completed operation.

Example

Example 1: In the following example, the database was enabled for text search in an older release. Calling the **SYSTS_UPGRADE_CATALOG** procedure upgrades the text search catalog tables, and if the procedure succeeds, the output parameter message indicating successful operation is returned to the caller.

```
CALL SYSPROC.SYSTS_UPGRADE_CATALOG('en_US',?)
```

Value of output parameters

Parameter Name : MESSAGE

Parameter Value : CIE0213W The Db2 Text Search Catalog has been upgraded to the current version. You will now need to update the text search index using the SYSPROC.SYSTS_UPGRADE_INDEX stored procedure.
index also needs to be updated by calling "SYSPROC.SYSTS_UPGRADE_INDEX".

Return Status = 0

Example 2: In the following example, the database was not enabled for Text Search in an older release before calling the procedure. A NULL value in the place *message_locale* means the a default locale of 'en_US' will be used. All error messages are returned in English.

```
CALL SYSPROC.SYSTS_UPGRADE_CATALOG('',?)
```

SQL20427N An error occurred during a text search administration procedure or command. The error message is "CIE0323E Specified or default database not enabled for text. ".
SQLSTATE=38H14

Example 3: In the following example, the text search catalog and the text search indexes are already upgraded to the current version.

```
CALL SYSPROC.SYSTS_UPGRADE_CATALOG('en_US',?)
```

Value of output parameters

Parameter Name : MESSAGE

Parameter Value : CIE0002I The Db2 Text Search release level is current for the database. The system has not been upgraded.

Return Status = 0

Usage notes

The **SYSTS_UPGRADE_CATALOG** procedure is integrated into the **DB2 UPGRADE DATABASE** command to perform the text search specific catalog upgrade. If the database upgrade fails to upgrade the text search catalog, the **SYSTS_UPGRADE_CATALOG** procedure must be executed separately by the user to complete the database upgrade.

To upgrade the text search index catalog, proceed as follows:

1. Make sure that the Text Search instance service is stopped.
2. Execute the **SYSTS_UPGRADE_CATALOG** procedure.

Note: Text Search administrative procedures use an existing connection to the database. The current transaction might be committed or rolled back depending on the completion of the procedures. As such, you might want to commit all transaction changes to avoid any unexpected impact from such a commit or rollback. One way to achieve this is to turn on **AUTOCOMMIT**.

3. The **SYSTS_UPGRADE_CATALOG** procedure attempts to populate the Text Search server information in the catalog. Review the SYSIBMTS.TSSERVER content and update the Text Search server information as necessary.
4. Upgrade the text search indexes by following the procedure outlined for the **SYSTS_UPGRADE_INDEX** procedure. The version value in the SYSIBMTS.TSDEFAULTS administrative view will not be updated until the **SYSTS_UPGRADE_INDEX** procedure is executed successfully.

SYSTS_UPGRADE_INDEX - Upgrade text search indexes

This procedure updates Text Search index information in database catalog tables and text search catalog tables.

Text search index collections are managed by the Text Search server. See Upgrading Text Search for more information.

Authorization

The privileges held by the authorization ID of the procedure must include the SYSTS_ADM role and the DBADM authority.

Default PUBLIC privilege

None

Syntax

►►SYSTS_UPGRADE_INDEX(—*message_locale*—,—*message*—)◄◄

The schema is SYSPROC.

Procedure parameters

message_locale

An input argument of type VARCHAR(33) that specifies the locale to be used for any error message returned. If the argument is null or an empty string, or the message files for the specified locale are not available on the server, 'en_US' is used.

message

An output argument of type VARCHAR(32K) that specifies a warning or informational message for a successfully completed operation.

Example

Example 1: In the following example, the database was enabled for text search in an older release and the procedure **SYSTS_UPGRADE_CATALOG** has already been completed successfully. The procedure **SYSTS_UPGRADE_INDEX** will complete the upgrade for the text index metadata in the database catalog. When the procedure succeeds, the output parameter message indicative of the successful operation is returned to the caller.

```
CALL SYSPROC.SYSTS_UPGRADE_INDEX('en_US', ?)
```

Parameter Name : MESSAGE

Parameter Value : CIE00001 Operation completed successfully.

Return Status = 0

Example 2: In the following example, the database was not enabled for text search in the older release of the product. If a NULL value is set for *message_locale*, it means the system locale will be used. If the system locale is not available then the default locale 'en_US' will be used.

```
CALL SYSPROC.SYSTS_UPGRADE_INDEX('', ?)
```

SQL20427N An error occurred during a text search administration procedure or command. The error message is "CIE0323E Specified or default database not enabled for text. ".
SQLSTATE=38H14

Example 3: In the following example, the Text Search catalog and the text search indexes were already upgraded to the current version.

```
CALL SYSPROC.SYSTS_UPGRADE_INDEX('en_US', ?)
```

Value of output parameters

Parameter Name : MESSAGE

Parameter Value : CIE00021 The Db2 Text Search release level is current for the database. The system has not been upgraded.

Return Status = 0

Example 4: In the following example, the Text Search catalog was not upgraded.

```
CALL SYSPROC.SYSTS_UPGRADE_INDEX('en_US', ?)
```

CIE0409E The Db2 Text Search catalog has not been upgraded to the current version.

Usage notes

- The **SYSTS_UPGRADE_INDEX** procedure is integrated into the **DB2 UPGRADE DATABASE** command to perform upgrades for text search indexes. When issuing the **DB2 UPGRADE DATABASE** command, this procedure will be executed as well. If the upgrade of text search indexes fails, the procedure must be run manually.
- The **SYSTS_UPGRADE_CATALOG** procedure should be executed before **SYSTS_UPGRADE_INDEX** procedure. The version value in the **SYSIBMTS.TSDEFAULTS** administrative view will not be updated until the **SYSTS_UPGRADE_INDEX** procedure is executed successfully.
- Text Search administrative procedures use an existing connection to the database. The current transaction might be committed or rolled back depending on the completion of the procedures. As such, you might want to commit all transaction changes to avoid any unexpected impact from such a commit or rollback. One way to achieve this is to turn on **AUTOCOMMIT**.

Workload management routines

This grouping of routines and views can be used to achieve effective management of work and workloads in your database.

WLM_CANCEL_ACTIVITY - Cancel an activity

This procedure cancels a given activity. If the cancel takes place, an error message will be returned to the application that submitted the activity that was cancelled.

Syntax

►►—WLM_CANCEL_ACTIVITY—(—*application_handle*—,—*uow_id*—,—*activity_id*—)————►◄

The schema is SYSPROC.

Procedure parameters

application_handle

An input argument of type BIGINT that specifies the application handle whose activity is to be cancelled. If the argument is null, no activity will be found and an SQL4702N with SQLSTATE 5U035 is returned.

uow_id

An input argument of type INTEGER that specifies the unit of work ID of the activity that is to be cancelled. If the argument is null, no activity will be found and an SQL4702N with SQLSTATE 5U035 is returned.

activity_id

An input argument of type INTEGER that specifies the activity ID which uniquely identifies the activity within the unit of work that is to be cancelled. If the argument is null, no activity will be found and an SQL4702N with SQLSTATE 5U035 is returned.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

- WLMADM authority

Default PUBLIC privilege

None

Example

An administrator can use the WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function to find the application handle, unit of work ID and activity ID of an activity. To cancel an activity with application handle 1, unit of work ID 2 and activity ID 3:

```
CALL WLM_CANCEL_ACTIVITY(1, 2, 3)
```

Usage notes

- If no activity can be found, an SQL4702N with SQLSTATE 5U035 is returned.
- If the activity cannot be cancelled because it not in the correct state (not initialized), an SQL4703N (reason code 1) with SQLSTATE 5U016 is returned.
- If the activity is successfully cancelled, an SQL4725N with SQLSTATE 57014 is returned to the cancelled application.
- If, at the time of the cancel, the coordinator is processing a request for a different activity or is idle, the activity is placed into CANCEL_PENDING state and will be cancelled when the coordinator processes the next request.

WLM_CAPTURE_ACTIVITY_IN_PROGRESS - Collect activity information for activities event monitor

The WLM_CAPTURE_ACTIVITY_IN_PROGRESS procedure gathers information about a specified activity and writes the information to the active activities event monitor.

The functionality of WLM_CAPTURE_ACTIVITY_IN_PROGRESS is identical to the functionality of MON_CAPTURE_ACTIVITY_IN_PROGRESS. Although the functionality for these two routines is the same, the authorization is not necessarily the same. See MON_CAPTURE_ACTIVITY_IN_PROGRESS for syntax, parameters, usage notes, and examples.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

WLM_COLLECT_STATS - Collect and reset workload management statistics

The WLM_COLLECT_STATS procedure gathers statistics for service classes, workloads, work classes, and threshold queues and writes them to the statistics event monitor.

This procedure also resets the statistics for service classes, workloads, work classes, and threshold queues.

If there is no active statistics event monitor, the procedure only resets the statistics.

The functionality of WLM_COLLECT_STATS is identical to the functionality of MON_COLLECT_STATS. Although the functionality for these two routines is the same, the authorization is not necessarily the same. See MON_COLLECT_STATS for syntax, parameters, usage notes, and examples.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

WLM_ALTER_MEMBER_SUBSET procedure - Alter a member subset object

The WLM_ALTER_MEMBER_SUBSET procedure alters an existing member subset database object.

Authorization

One of the following authorizations is required:

- DBADM authority
- WLMADM authority

Default PUBLIC privilege

None

Syntax

►►—WLM_ALTER_MEMBER_SUBSET—(—*subset_name*—,—*subset_attributes*—,—*member_list*—)————►◄

The schema is SYSPROC.

Routine parameters

subset_name

An input argument of type VARCHAR(128) that specifies the name of the member subset object to be altered. This is a one-part name. It is an SQL identifier (ordinary or delimited). The name must identify a member subset which already exists in the current database (SQLSTATE 42704). This argument cannot be NULL (SQLSTATE 42815).

subset_attributes

An input argument of type CLOB(8K) that enables you to alter one or more member subset attributes. This parameter defaults to NULL. Subset attributes are specified as name value pairs using the format:

```
<subset attribute name tag>value</subset attribute name tag>
```

Each subset attribute can be specified a maximum of one time (SQLSTATE 42613). Subset attribute names are case sensitive. The subset attribute values are case insensitive unless otherwise noted. The available subset attribute name tags are as follows:

'<enabled>value</enabled>'

Specifies whether this member subset will be considered when a connection request is assigned to a member subset. This value can be either "yes" or "no". If not specified the current value is not altered.

'<databaseAlias>value</databaseAlias>'

Replace the database alias that a client must connect to in order to be assigned to this member subset. The name must conform to the naming conventions for local database aliases (SQLSTATE 2E000).

If the existing database alias was cataloged by a member subset operation it is uncataloged by this command.

'<catalogDatabaseAlias>value</catalogDatabaseAlias>'

Specifies whether the procedure should catalog the new database alias specified in the databaseAlias attribute. This value can be either "yes" or "no". When this attribute is specified a new database alias attribute must also be provided (SQLSTATE 530AA).

If the value of this attribute is "yes" the new database alias specified in the databaseAlias attribute must not already be cataloged (SQL1005N). If the value of this attribute is "no" the new database alias specified in the databaseAlias attribute must already be cataloged (SQLSTATE 42705).

'<memberPriorityBasis>value</memberPriorityBasis>'

Specifies what information should be used when calculating the relative member priorities in the member subsets server list. This value can be either "load" or "equalPriority". If not specified, the current value is not altered.

The value "load" indicates that the member priorities in the member subsets server list will be based on the relative load of each member. The relative load is calculated based on the average CPU load and average page swap rate. This value can be specified only in a Db2 pureScale instance (SQLSTATE 530AA).

The value "equalPriority" indicates that the member priority in the member subsets server list for each active member will be 101.

'<inclusiveSubset>value</inclusiveSubset>'

Specifies whether all members in the instance will be included in the member subset's server list for high availability (HA) purposes when no other member of the subset is online.

This attribute can be specified only in a Db2 pureScale instance (SQLSTATE 530AA). When specified the value can be either "yes" or "no". If not specified, the current value is not altered.

'<alternateServer>value</alternateServer>'

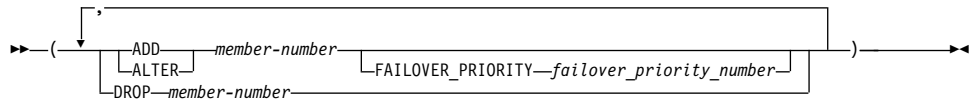
Specifies whether the host name and port cataloged by the UPDATE ALTERNATE SERVER statement will be included in the member subset's server list for HA purposes when no other member of the subset is online. This value can be either "yes" or "no". If not specified, it defaults to "no".

'<comment>value</comment>'

Specifies a comment on the member subset. The comment specified replaces the value in the REMARKS column of the SYSCAT.MEMBERSUBSETS catalog view for the row that describes the member subset. The comment value can be any character string constant of up to 254 bytes.

member_list

An input argument of type VARCHAR(32000) that enables you to specify a list of members to add to, alter, or drop from, the member subset. This parameter defaults to NULL. The list of members are specified as comma-separated numerical values by using the following syntax:



Each member number can be specified a maximum of one time (SQLSTATE 42613). If the ADD token is specified, the member must not already be included in the member subset (SQLSTATE 42613). If the ALTER or DROP token is specified, the member must already be included in the member subset. The member number must be defined in the instance (SQLSTATE 42729). The last primary member in a member subset cannot be dropped (SQLSTATE 530AB).

failover_priority_number

In a Db2 pureScale environment, the *failover_priority_number* argument can be specified with *member_list*. Use *failover_priority_number* with *member_list* to specify a failover priority for the members that are being added or altered in the member subset. The parameter defaults to 0. Each subset must have at least one member with FAILOVER_PRIORITY set to 0. The value for *failover_priority_number* can be 0 - 254 (SQLSTATE 42815).

Usage notes

- As an autonomous procedure, WLM_ALTER_MEMBER_SUBSET executes in its own transaction scope.
- When WLM_ALTER_MEMBER_SUBSET successfully completes, it will commit the work performed within the procedure, but if it is unsuccessful, the procedure rolls back any work performed.

Examples

1. Add member 3 to the member subset MY_SUBSET.
`CALL SYSPROC.WLM_ALTER_MEMBER_SUBSET('MY_SUBSET', NULL, '(ADD 3)')`
2. Drop member 2 from the member subset MY_SUBSET.
`CALL SYSPROC.WLM_ALTER_MEMBER_SUBSET('MY_SUBSET', NULL, '(DROP 2)')`
3. Change the member priority basis to equal priorities.
`CALL SYSPROC.WLM_ALTER_MEMBER_SUBSET('MY_SUBSET',
'<memberPriorityBasis>equalPriority</memberPriorityBasis>', NULL)`
4. Change the member subset to an exclusive subset.

```
CALL SYSPROC.WLM_ALTER_MEMBER_SUBSET('MY_SUBSET',
    '<inclusiveSubset>no</inclusiveSubset>', NULL)
```

5. Replace the current database alias subset attribute from MY_SUBSET with NEWALIAS.

```
CALL SYSPROC.WLM_ALTER_MEMBER_SUBSET('MY_SUBSET',
    '<databaseAlias>NEWALIAS</databaseAlias>', NULL)
```

6. Add member 4 with FAILOVER_PRIORITY 1 to the member subset MY_SUBSET.

```
CALL SYSPROC.WLM_ALTER_MEMBER_SUBSET('MY_SUBSET', NULL, '(ADD 4 FAILOVER_PRIORITY 1)')
```

WLM_CREATE_MEMBER_SUBSET procedure - Create a member subset object

The WLM_CREATE_MEMBER_SUBSET procedure creates a member subset database object.

Authorization

One of the following authorizations is required:

- DBADM authority
- WLMADM authority

Default PUBLIC privilege

None

Syntax

```
►►—WLM_CREATE_MEMBER_SUBSET—(—subset_name—,—subset_attributes—,—member_list—)————►◄
```

The schema is SYSPROC.

Routine parameters

subset_name

An input argument of type VARCHAR(128) that specifies the name of the member subset object to be created. This is a one-part name. It is an SQL identifier (ordinary or delimited). The name must not identify a member subset which already exists in the current database (SQLSTATE 42710). The name must not begin with the characters 'SYS' (SQLSTATE 42939). This argument cannot be NULL (SQLSTATE 42815).

subset_attributes

An input argument of type CLOB(8K) that enables you to specify one or more member subset attributes. This parameter cannot be NULL (SQLSTATE 42815). Subset attributes are specified as name value pairs using the format:

```
<subset attribute name tag>value</subset attribute name tag>
```

Each subset attribute can be specified a maximum of one time (SQLSTATE 42613). Subset attribute names are case sensitive. The subset attribute values are case insensitive unless otherwise noted. The available subset attribute name tags are as follows:

'<enabled>value</enabled>'

Specifies whether this member subset will be considered when a connection request is assigned to a member subset. This value can be either "yes" or "no". If not specified it defaults to "yes".

'<databaseAlias>value</databaseAlias>'

Specifies a database alias name that a client connection must connect to in order to be assigned to this member subset. The name must conform to naming conventions for local database aliases (SQLSTATE 2E000). This attribute is required (SQLSTATE 42815).

'<catalogDatabaseAlias>value</catalogDatabaseAlias>'

Specifies whether the procedure should catalog a database alias if that is one of the specified subset attributes. This value can be either "yes" or "no". If not specified it defaults to "yes".

If the value of this attribute is "yes" the database alias specified in the databaseAlias attribute must not already be cataloged (SQL1005N). If the value of this attribute is "no" the database alias specified in the databaseAlias attribute must already be cataloged (SQLSTATE 42705).

'<memberPriorityBasis>value</memberPriorityBasis>'

Specifies what information should be used when calculating the relative member priorities in the member subsets server list. This value can be either "load" or "equalPriority". If not specified, it defaults to "load" in a Db2 pureScale instance and "equalPriority" in a partitioned database or single member environment.

The value "load" indicates that the member priorities in the member subsets server list will be based on the relative load of each member. The relative load is calculated based on the average CPU load and average page swap rate. This value can be specified only in a Db2 pureScale instance (SQLSTATE 530AA).

The value "equalPriority" indicates that the member priority in the member subsets server list for each active member will be 100.

'<inclusiveSubset>value</inclusiveSubset>'

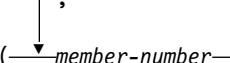
Specifies whether all members in the instance will be included in the member subset's server list for high availability (HA) purposes when no other member of the subset is online. When specified the value can be either "yes" or "no". If not specified, it defaults to "yes" in a Db2 pureScale instance and "no" in a partitioned database or single member environment. This value can be specified only in a Db2 pureScale instance (SQLSTATE 530AA).

'<alternateServer>value</alternateServer>'

Specifies whether the host name and port cataloged by the UPDATE ALTERNATE SERVER statement will be included in the member subset's server list for HA purposes when no other member of the subset is online. This value can be either "yes" or "no". If not specified, it defaults to "no".

member_list

An input argument of type VARCHAR(32000) that enables you to specify a list of members to include in the member subset. This parameter cannot be NULL (SQLSTATE 42815). The list of members are specified as comma separated numerical values using the following syntax:

→→ () →→

Each member number can be specified a maximum of one time (SQLSTATE

42613). The member number must be defined in the instance (SQLSTATE 42729).

Usage notes

- As an autonomous procedure, WLM_CREATE_MEMBER_SUBSET executes in its own transaction scope.
- When WLM_CREATE_MEMBER_SUBSET successfully completes, it will commit the work performed within the procedure, but if it is unsuccessful, the procedure rolls back any work performed.
- When a database connection is established, the database manager looks for a matching member subset based on the database alias subset attribute. If a matching member subset is found, the connection is associated with the member subset.

Examples

1. Create a member subset in a Db2 pureScale instance for connections to the database alias PSALIAS. The PSALIAS database alias does not exist and will be cataloged by the WLM_CREATE_MEMBER_SUBSET procedure.

```
CALL SYSPROC.WLM_CREATE_MEMBER_SUBSET(
    'PURESCALE_SUBSET',
    '<databaseAlias>PSALIAS</databaseAlias>',
    '(0, 1)' )
```

2. Create a member subset in a partitioned database environment for connections to the database alias DPFALIAS. The DPFALIAS database alias does not exist and will be cataloged by the WLM_CREATE_MEMBER_SUBSET procedure.

The server list for the member subset is to include members 100 and 200 only.

```
CALL SYSPROC.WLM_CREATE_MEMBER_SUBSET(
    'DPF_SUBSET',
    '<databaseAlias>DPFALIAS</databaseAlias>',
    '(100, 200)' )
```

3. Create a member subset in a Db2 pureScale instance for connections to the database alias OLTPDB. The OLTPDB database alias exists prior to executing the WLM_CREATE_MEMBER_SUBSET procedure.

The server list for the member subset is to include members 1 and 2 for workload balancing usage and all other members for automatic client rerouting purposes.

```
CALL SYSPROC.WLM_CREATE_MEMBER_SUBSET(
    'OLTP_SUBSET',
    '<catalogDatabaseAlias>no</catalogDatabaseAlias>'
    '<databaseAlias>OLTPDB</databaseAlias>',
    '(1, 2)' )
```

4. Create a member subset in a Db2 pureScale instance for connections to the database alias EQUALDB. The EQUALDB database alias does not exist and will be cataloged by the WLM_CREATE_MEMBER_SUBSET procedure.

The server list for the member subset is to include members 0 and 1 only. The member priority in the server list are to be equivalent for all active members.

```
CALL SYSPROC.WLM_CREATE_MEMBER_SUBSET(
    'EQUAL_SUBSET',
    '<memberPriorityBasis>equalPriority</memberPriorityBasis>'
    '<inclusiveSubset>no</inclusiveSubset>'
    '<databaseAlias>EQUALDB</databaseAlias>',
    '(0, 1)' )
```

WLM_DROP_MEMBER_SUBSET procedure - Drop a member subset object

The WLM_DROP_MEMBER_SUBSET procedure drops an existing member subset database object.

Authorization

One of the following authorizations is required:

- DBADM authority
- WLMADM authority

Default PUBLIC privilege

None

Syntax

►►—WLM_DROP_MEMBER_SUBSET—(—*subset_name*—)—————►►

The schema is SYSPROC.

Routine parameters

subset_name

An input argument of type VARCHAR(128) that specifies the name of the member subset object to be dropped. This is a one-part name. It is an SQL identifier (ordinary or delimited). The name must identify a member subset which already exists in the current database (SQLSTATE 42704). This argument cannot be NULL (SQLSTATE 42815).

Usage notes

- As an autonomous procedure, WLM_DROP_MEMBER_SUBSET executes in its own transaction scope.
- When WLM_DROP_MEMBER_SUBSET successfully completes, it will commit the work performed within the procedure, but if it is unsuccessful, the procedure rolls back any work performed.

Example

Drop the member subset MY_SUBSET.

```
CALL SYSPROC.WLM_DROP_MEMBER_SUBSET( 'MY_SUBSET' )
```

WLM_GET_CONN_ENV - get settings for activity data collection for a connection

The WLM_GET_CONN_ENV table function returns for a particular connection the values of settings that control collection of activity data and section actuals.

You can use this table function to check the current values of the settings applied by the WLM_SET_CONN_ENV stored procedure.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine

- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Syntax

►►—WLM_GET_CONN_ENV—(—*application_handle*—)————►►

Parameters

application_handle

An input argument of type BIGINT that specifies the application handle for the connection for which information is to be returned. You can use a value of NULL to indicate the connection on which the procedure was invoked.

Information returned

Table 210. Information returned for WLM_GET_CONN_ENV

| Column name | Data type | Description |
|--------------------|-----------|--|
| APPLICATION_HANDLE | BIGINT | application_handle - Application handle |
| DETAILS | BLOB(8M) | XML document that contains connection environment details. See Table 211 for a description of the elements in this document. |

Detailed settings returned

Table 211. Detailed metrics returned for WLM_GET_CONN_ENV

| Element name | Data type | Description or corresponding monitor element |
|---------------------|----------------|---|
| collectactdata | xs:string(255) | Specifies what kind of activity data is being collected, if any. Possible values are: <ul style="list-style-type: none"> • NONE • WITHOUT DETAILS • WITH DETAILS • WITH DETAILS, SECTION • WITH DETAILS, SECTION AND VALUES • WITH DETAILS AND VALUES See information about the WLM_SET_CONN_ENV procedure for details about these options. |
| collectactpartition | xs:string(255) | Specifies where activity data is being collected. Possible values are: <ul style="list-style-type: none"> • COORDINATOR • ALL See information about the WLM_SET_CONN_ENV procedure for details about these options. |

Table 211. Detailed metrics returned for WLM_GET_CONN_ENV (continued)

| Element name | Data type | Description or corresponding monitor element |
|-----------------------|----------------|---|
| collectsectionactuals | xs:string(255) | Specifies whether section actuals are being collected.
Possible values include: <ul style="list-style-type: none"> NONE BASE See information about the WLM_SET_CONN_ENV procedure for details about these options. |

Usage notes

The WLM_GET_CONN_ENV table function provides maximum flexibility for formatting output because it returns WLM environment information for a connection as an XML document. The output can be parsed directly by an XML parser, or it can be converted to relational format by the XMLTABLE function.

The schema for the XML document that is returned in the DETAILS column is available in the file sqllib/misc/DB2MonRoutines.xsd. Further details can be found in the file sqllib/misc/DB2MonCommon.xsd.

Example

The following query checks whether activities are being collected for the current connection.

```
SELECT application_handle,
       xmlparse(document details preserve whitespace)
FROM TABLE (
  WLM_GET_CONN_ENV(
    cast(NULL as bigint))
  ) connenv
```

The following is an example of output from this query.

```
APPLICATION_HANDLE  DETAILS
-----
7 <wlm_conn_env
  xmlns=http://www.ibm.com/xmlns/prod/db2/mon
  release="9070100">
  <collectactdata>NONE</collectactdata>
  <collectactpartition>COORDINATOR</collectactpartition>
  <collectsectionactuals>NONE</collectsectionactuals>
</wlm_conn_env>
```

WLM_GET_QUEUE_STATS table function - Return threshold queue statistics

The WLM_GET_QUEUE_STATS function returns basic statistics for one or more threshold queues on all active members. This function returns one row of statistics for each threshold queue.

The functionality of WLM_GET_QUEUE_STATS is identical to the functionality of MON_GET_QUEUE_STATS. Although the functionality for these two routines is the same, the authorization is not necessarily the same. See MON_GET_QUEUE_STATS for syntax, parameters, information returned, usage notes, and examples.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

WLM_GET_SERVICE_CLASS_AGENTS table function - list agents running in a service class

The WLM_GET_SERVICE_CLASS_AGENTS function returns the list of agents, fenced mode processes (db2fmp processes), and system entities on a specified member that are running in a specified service class or on behalf of a specified application.

The system entities that WLM_GET_SERVICE_CLASS_AGENTS returns are non-agent threads and processes, such as page cleaners and prefetchers.

The functionality of WLM_GET_SERVICE_CLASS_AGENTS is identical to the functionality of MON_GET_AGENT. Although the functionality for these two routines is the same, the authorization is not necessarily the same. See MON_GET_AGENT for syntax, parameters, information returned, usage notes, and examples.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES - list workload occurrences

The WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES function returns the list of all workload occurrences running in a specified service class on a particular member. A workload occurrence is a specific database connection whose attributes match the definition of a workload and hence is associated with or assigned to the workload.

Refer to Table 212 on page 1066 for a complete list of information that can be returned.

Syntax

►—WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES—(—*service_superclass_name*—,——►
►—*service_subclass_name*—, —*member*—)——►

The schema is SYSPROC.

Table function parameters

service_superclass_name

An input argument of type VARCHAR(128) that specifies the name of a service superclass in the currently connected database. If the argument is null or an empty string, the data is retrieved for all the superclasses in the database that match the values of the other parameters.

service_subclass_name

Target service subclass for the workload occurrence. Any work submitted by this workload occurrence will run in this service subclass under the target service superclass with the exception of activities that are mapped, or remapped, to a different subclass.

member

An input argument of type INTEGER that specifies the number of a member in the same instance as the currently connected database. Specify -1 for the current database member, or -2 for all active database members. If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Example

If an administrator wants to see what workload occurrences are running on the system as a whole, the administrator can call the WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES function by specifying a null value or an empty string for *service_superclass_name* and *service_subclass_name* and -2 for *member*:

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,  
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,  
       SUBSTR(CHAR(MEMBER),1,4) AS MEMB,  
       SUBSTR(CHAR(COORD_MEMBER),1,4) AS COORDMEMB,  
       SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHNDL,  
       SUBSTR(WORKLOAD_NAME,1,22) AS WORKLOAD_NAME,  
       SUBSTR(CHAR(WORKLOAD_OCCURRENCE_ID),1,6) AS WLO_ID  
FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES
```

```

(CAST(NULL AS VARCHAR(128)), CAST(NULL AS VARCHAR(128)), -2))
AS SCINFO
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, MEMB, APPHNDL,
        WORKLOAD_NAME, WLO_ID

```

If the system has four database members and is currently running two workloads, the previous query produces results such as the following ones:

```

SUPERCLASS_NAME  SUBCLASS_NAME  MEMB COORDMEMB ...
-----
SYSDEFAULTMAINTENAN SYSDEFAULTSUBCLASS 0    0    ...
SYSDEFAULTSYSTEMCLA SYSDEFAULTSUBCLASS 0    0    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0    0    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 0    0    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1    0    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 1    0    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2    0    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 2    0    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3    0    ...
SYSDEFAULTUSERCLASS SYSDEFAULTSUBCLASS 3    0    ...
... APPHNDL WORKLOAD_NAME          WLO_ID
... -----
... -      -                      -
... -      -                      -
... 1      SYSDEFAULTUSERWORKLOAD 1
... 2      SYSDEFAULTUSERWORKLOAD 2
... 1      SYSDEFAULTUSERWORKLOAD 1
... 2      SYSDEFAULTUSERWORKLOAD 2
... 1      SYSDEFAULTUSERWORKLOAD 1
... 2      SYSDEFAULTUSERWORKLOAD 2
... 1      SYSDEFAULTUSERWORKLOAD 1
... 2      SYSDEFAULTUSERWORKLOAD 2

```

Usage note

The parameters are, in effect, ANDed together. That is, if you specify conflicting input parameters, such as a service superclass SUP_A and a subclass SUB_B such that SUB_B is not a subclass of SUP_A, no rows are returned.

Note: Statistics reported for the workload occurrence (for example, coord_act_completed_total) are reset at the beginning of each unit of work when they are combined with the corresponding workload statistics.

Information returned

Table 212. Information returned for WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES

| Column name | Data type | Description |
|-------------------------|--------------|--|
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name monitor element |
| SERVICE_SUBCLASS_NAME | VARCHAR(128) | service_subclass_name - Service subclass name monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| COORD_PARTITION_NUM | SMALLINT | coord_partition_num - Coordinator partition number monitor element |
| APPLICATION_HANDLE | BIGINT | application_handle - Application handle monitor element |
| WORKLOAD_NAME | VARCHAR(128) | workload_name - Workload name monitor element |

Table 212. Information returned for WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES (continued)

| Column name | Data type | Description |
|---------------------------|--------------|--|
| WORKLOAD_OCCURRENCE_ID | INTEGER | workload_occurrence_id - Workload occurrence identifier monitor element |
| UOW_ID | INTEGER | uow_id - Unit of work ID monitor element |
| WORKLOAD_OCCURRENCE_STATE | VARCHAR(32) | workload_occurrence_state - Workload occurrence state monitor element |
| SYSTEM_AUTH_ID | VARCHAR(128) | system_auth_id - System authorization identifier monitor element |
| SESSION_AUTH_ID | VARCHAR(128) | session_auth_id - Session authorization ID monitor element |
| APPLICATION_NAME | VARCHAR(128) | appl_name - Application name monitor element |
| CLIENT_WRKSTNNAME | VARCHAR(255) | client_wrkstnname - Client workstation name monitor element |
| CLIENT_ACCTNG | VARCHAR(255) | client_acctng - Client accounting string monitor element |
| CLIENT_USER | VARCHAR(255) | Current value of the CLIENT_USERID special register for this workload occurrence. |
| CLIENT_APPLNAME | VARCHAR(255) | client_applname - Client application name monitor element |
| COORD_ACT_COMPLETED_TOTAL | INTEGER | coord_act_completed_total - Coordinator activities completed total monitor element |
| COORD_ACT_ABORTED_TOTAL | INTEGER | coord_act_aborted_total - Coordinator activities aborted total monitor element |
| COORD_ACT_REJECTED_TOTAL | INTEGER | coord_act_rejected_total - Coordinator activities rejected total monitor element |
| CONCURRENT_ACT_TOP | INTEGER | concurrent_act_top - Concurrent activity top monitor element |
| ADDRESS | VARCHAR(255) | address - IP address from which the connection was initiated |
| APPL_ID | VARCHAR(128) | appl_id - Application ID |
| MEMBER | SMALLINT | member - Database member monitor element |
| COORD_MEMBER | SMALLINT | coord_member - Coordinator member monitor element |

WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses

The WLM_GET_SERVICE_SUBCLASS_STATS function returns basic statistics for one or more service subclasses.

The functionality of WLM_GET_SERVICE_SUBCLASS_STATS is identical to the functionality of MON_GET_SERVICE_SUBCLASS_STATS. Although the functionality for these two routines is the same, the authorization is not necessarily the same. See MON_GET_SERVICE_SUBCLASS_STATS for syntax, parameters, information returned, usage notes, and examples.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine

- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

WLM_GET_SERVICE_SUPERCLASS_STATS - Return statistics of service superclasses

The WLM_GET_SERVICE_SUPERCLASS_STATS function returns basic statistics for one or more service superclasses.

The functionality of WLM_GET_SERVICE_SUPERCLASS_STATS is identical to the functionality of MON_GET_SERVICE_SUPERCLASS_STATS. Although the functionality for these two routines is the same, the authorization is not necessarily the same. See MON_GET_SERVICE_SUPERCLASS_STATS for syntax, parameters, information returned, usage notes, and examples.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

WLM_GET_WORK_ACTION_SET_STATS - Return work action set statistics

The WLM_GET_WORK_ACTION_SET_STATS function returns the statistics for a work action set.

The functionality of WLM_GET_WORK_ACTION_SET_STATS is identical to the functionality of MON_GET_WORK_ACTION_SET_STATS. Although the functionality for these two routines is the same, the authorization is not necessarily the same. See MON_GET_WORK_ACTION_SET_STATS for syntax, parameters, information returned, usage notes, and examples.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES - Return a list of activities

The WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES function returns the list of all activities that were submitted by the specified application on the specified member and have not yet been completed.

The functionality of WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES is identical to the functionality of MON_GET_ACTIVITY. Although the functionality for these two routines is the same, the authorization is not necessarily the same. See MON_GET_ACTIVITY for syntax, parameters, information returned, usage notes, and examples.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

WLM_GET_WORKLOAD_STATS table function - Return workload statistics

The WLM_GET_WORKLOAD_STATS function returns one row of workload statistics for every combination of workload name and database member number.

The functionality of WLM_GET_WORKLOAD_STATS is identical to the functionality of MON_GET_WORKLOAD_STATS. Although the functionality for these two routines is the same, the authorization is not necessarily the same. See MON_GET_WORKLOAD_STATS for syntax, parameters, information returned, usage notes, and examples.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

WLM_SET_CLIENT_INFO procedure - Set client information

The WLM_SET_CLIENT_INFO procedure sets client information associated with the current connection at the database server.

By using this procedure, you can set the client's user ID, application name, workstation name, accounting information, or workload information at the database server. Calling this procedure changes the stored values of the relevant transaction processor (TP) monitor client information fields and special register settings for this connection.

The client information fields are used at the database server for determining the identity of the application or user currently using the connection. The client information fields for a connection are considered during database workload evaluation and also displayed in any database audit records or application snapshots generated for this connection.

Unlike the sqleseti API, this procedure does not set client information at the client but instead sets the corresponding client attributes on the database server. Therefore, you cannot use the sqleqry API to query the client information that is set at the database server using this procedure. If an application uses the sqleseti API to change the client information, the new values will change the setting at the database server. If the sqleseti API is used to change either the user ID or the application name without changing the accounting information, the accounting information at the database server will also be reset to the value of the accounting information at the client.

The data values provided with the procedure are converted to the appropriate database code page before being stored in the related TP monitor fields or special registers. Any data value which exceeds the maximum supported size after conversion to the database code page is truncated before being stored at the server. The truncated values are returned by both the TP monitor fields and the special registers when those stored values are queried.

The WLM_SET_CLIENT_INFO procedure is not under transaction control, and client information changes made by the procedure are independent of committing or rolling back units of work. However, because workload reevaluation occurs at the beginning of the next unit of work for each application, you must issue either a COMMIT or a ROLLBACK statement to make client information changes effective.

Syntax

```
►►—WLM_SET_CLIENT_INFO—(—client_userid—,—client_wrkstnname—,——————►  
►—client_applname—,—client_acctstr—,—client_workload—)——————►◄
```

The schema is SYSPROC.

Procedure parameters

client_userid

An input argument of type VARCHAR(255) that specifies the user ID for the client. If you specify NULL, the value remains unchanged. If you specify an empty string, which is the default value, the user ID for the client is reset to the default value, which is blank.

client_wrkstnname

An input argument of type VARCHAR(255) that specifies the workstation name for the client. If you specify NULL, the value remains unchanged. If you specify an empty string, which is the default value, the workstation name for the client is reset to the default value, which is blank.

client_applname

An input argument of type VARCHAR(255) that specifies the application name for the client. If you specify NULL, the value remains unchanged. If you specify an empty string, which is the default value, the application name for the client is reset to the default value, which is blank.

client_acctstr

An input argument of type VARCHAR(255) that specifies the accounting string for the client. If you specify NULL, the value remains unchanged. If you specify an empty string, which is the default value, the accounting string for the client is reset to the default value, which is blank.

client_workload

An input argument of type VARCHAR(255) that specifies the workload assignment mode for the client. If you specify NULL, the value remains unchanged. The values are as follows:

SYSDEFAULTADMWORKLOAD

Specifies that the database connection will be assigned to SYSDEFAULTADMWORKLOAD, enabling users with ACCESSCTRL, DATAACCESS, DBADM, SECADM, or WLMADM authority to bypass the normal workload evaluation.

AUTOMATIC

Specifies that the database connection will be assigned to a workload chosen by the workload evaluation that is performed automatically by the server.

Note: The *client_workload* argument is case sensitive.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Examples

The following procedure call sets the user ID, workstation name, application name, accounting string, and workload assignment mode for the client:

```
CALL SYSPROC.WLM_SET_CLIENT_INFO('db2user', 'machine.torolab.ibm.com',  
    'auditor', 'Accounting department', 'AUTOMATIC')
```

The following procedure call sets the user ID to db2user2 for the client without setting the other client attributes:

```
CALL SYSPROC.WLM_SET_CLIENT_INFO('db2user2', NULL, NULL, NULL, NULL)
```

The following procedure call resets the user ID for the client to blank without modifying the values of the other client attributes:

```
CALL SYSPROC.WLM_SET_CLIENT_INFO('', NULL, NULL, NULL, NULL)
```

Usage notes

If the input that you specify for any procedure parameter exceeds its specified field length, the input field is truncated and the procedure runs using the truncated inputs.

Input fields containing single quotations are not supported and result in an error.

WLM_SET_CONN_ENV - enable collection of activity data and measurement of section actuals

The WLM_SET_CONN_ENV procedure enables for a particular connection the collection of activity data and measurement of section actuals (runtime statistics measured during section execution).

Once applied, the settings made by the WLM_SET_CONN_ENV procedure continue to apply until explicitly overwritten by another call to the WLM_SET_CONN_ENV procedure, or until the connection is closed. After the connection is closed, any new connection that reuses the same application handle does not inherit the settings of the previous connection to use that application handle.

►►—WLM_SET_CONN_ENV—(—*application_handle*—,—*settings*—)—————►◄

The schema is SYSPROC.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Parameters

application_handle

An input argument of type BIGINT that specifies the application handle whose connection environment is to be modified. The application handle specified must refer to an existing application (otherwise, SQLSTATE 5U002 is returned).

You can use a value of NULL to indicate that the connection whose environment is to be changed is the connection on which the procedure was invoked.

settings

An input argument of type CLOB(8K) that enables you to specify one or more monitor settings. Settings are specified as name value pairs using the format:

`<setting name tag>value</setting name tag>`

Each setting can be specified a maximum of one time. Setting names are case sensitive. A change to a setting takes effect on the next statement executed; it has no effect on statements already in progress.

The available setting name tags are as follows.

- '`<collectactdata>value</collectactdata>`'

Specifies what activity data should be collected by the activity event monitor. The possible values are (variations in spaces between words are supported):

| Value | Description |
|----------------------------------|---|
| NONE | Activity data should not be collected |
| WITHOUT DETAILS | Data about each activity is sent to any active activities event monitor when the activity completes execution. Details about statement, compilation environment, and section environment data are not sent. |
| WITH DETAILS | Statement and compilation environment data is sent to any active activities event monitor, for those activities that have them. Section environment data is not sent. |
| WITH DETAILS, SECTION | Statement, compilation environment, section environment data, and section actuals are sent to any active activities event monitor, for those activities that have them.

For section actuals to be collected, either <i>collectsectionactuals</i> must be set to BASE or the section_actuals database configuration parameter must be set to BASE. Section actuals are collected on any partition where the activity data is collected. |
| WITH DETAILS, SECTION AND VALUES | Statement, compilation environment, section environment data, section actuals, and input data values are sent to any active activities event monitor, for those activities that have them.

For section actuals to be collected, either <i>collectsectionactuals</i> must be set to BASE or the section_actuals database configuration parameter must be set to BASE. Section actuals are collected on any partition where the activity data is collected. |

| Value | Description |
|-------------------------|---|
| WITH DETAILS AND VALUES | Statement, compilation environment, and input data values are sent to any active activities event monitor, for those activities that have them. Section environment data is not sent. |

- '<collectactpartition>COORDINATOR</collectactpartition>' or '<collectactpartition>ALL</collectactpartition>'

Specifies where activity data is collected, either just at the coordinator partition or at all partitions. If *collectactpartition* is not specified, the connection maintains its previous value for *collectactpartition* which by default is COORDINATOR.
- '<collectsectionactuals>NONE</collectsectionactuals>' or '<collectsectionactuals>BASE</collectsectionactuals>'

Section actuals are collected if *collectsectionactuals* is set to BASE.

Example

The following examples both enable activity collection, without details, on the coordinator partition for the current connection:

```
CALL WLM_SET_CONN_ENV(NULL, '<collectactdata>WITHOUT DETAILS</collectactdata>')
CALL WLM_SET_CONN_ENV(NULL, '<collectactdata>WITHOUT
DETAILS</collectactdata><collectactpartition>COORDINATOR
</collectactpartition>')
```

The next example enables collection of activity data with section environment data and section actuals , but no data values, on all partitions for the current connection:

```
CALL WLM_SET_CONN_ENV(NULL, '<collectactdata>WITH DETAILS, SECTION
</collectactdata><collectactpartition>ALL</collectactpartition>')
```

The following example disables collection of activity data for the current connection.

```
CALL WLM_SET_CONN_ENV(NULL, '<collectactdata>NONE</collectactdata>')
```

Usage notes

The *collectactdata* setting only controls activity data collection at the connection level . An activity might have multiple activity data collection controls applied to it, for example, the connection might be mapped to a service class where the COLLECT ACTIVITY DATA clause has been applied. In a situation where multiple activity data collection controls are applied, the effective setting is the combination of all the settings. For example:

1. The connection level control is activity data without details.
2. The workload control is none.
3. The service class control is activity data with details and values.
4. When the activity completes execution, detailed information about the activity plus data values is sent to any active event monitors.

If a setting is not specified in the input of the WLM_SET_CONN_ENV procedure, it is not altered in the connection environment.

The effective setting for the collection of section actuals is the combination of the *collectsectionactuals* setting and the **section_actuals** database configuration parameter. For example, if *collectsectionactuals* is set to BASE and the **section_actuals** database configuration parameter value is NONE, the effective setting for the collection of section actuals is BASE (and vice versa).

Automatic client rerouting cannot be performed for a connection when activity data and section actuals are being collected (when *collectactdata* is set to any value other than NONE).

Miscellaneous routines and views

This grouping of routines and views can be used to accomplish various, category-independent tasks.

ALTOBJ

The ALTOBJ procedure parses an input CREATE TABLE statement that serves as the target data definition language (DDL) for an existing table that is to be altered.

The procedure backs up the data of the table being altered, then drops the original table and creates a new version using the DDL statement; the final step loads the stored data back into the new table.

This procedure supports the following alter table operations and maintains recoverable dependencies:

- Renaming a column
- Increasing or decreasing the size of a column
- Altering a column type and transforming existing data using scalar functions
- Changing the precision or the scale of decimal values
- Changing the default value of a column
- Changing the nullability attribute of a column to nullable
- Dropping a column

Syntax

►►ALTOBJ—(*—exec-mode—*,*—sql-stmt—*,*—alter-id—*,*—msg—*)—————►►

The schema is SYSPROC.

Procedure parameters

exec-mode

An input argument of type VARCHAR(30) that specifies one of the following execution modes:

'GENERATE'

Specifies that all the scripts required by the VALIDATE, APPLY, and UNDO modes are to be generated.

'VALIDATE'

Specifies that the statement syntax is to be validated. This option also generates a script to manage the processing of related objects and relationships for the table that is to be altered.

'APPLY_CONTINUE_ON_ERROR' or 'APPLY_STOP_ON_ERROR'

Specifies that a script to manage the processing of related objects and

relationships for the table that is to be altered is to be generated. Data from the original table is to be exported, transformed, and used to populate the new table.

'UNDO'

Specifies that any changes made by the alter table operation are to be undone, in case a rollback operation cannot recover errors that might have occurred. This mode is only possible if the original table and any generated scripts have not been deleted. In this mode, the *alter-id* parameter must be specified as generated in a previous ALTOBJ operation and cannot have a value of -1, otherwise SQL0443N is returned.

'FINISH'

Specifies that the renamed original table is to be dropped.

sql-stmt

An input argument of type VARCHAR(2048) that specifies a CREATE TABLE statement that will be used as a template for altering an existing table. When *exec-mode* is 'GENERATE', *sql-stmt* must not be the null value. Otherwise, *sql-stmt* can be the null value, but only if *alter-id* is not -1.

alter-id

An input and output argument of type INTEGER that identifies all of the statements that are generated by this call. If -1 is specified, a new identifier will be generated and returned to the caller. Any existing statements identified by the specified integer are overwritten.

msg

An output argument of type VARCHAR(2048) containing an SQL query that you can execute to display all of the SQL statements generated for or used by the alter table process under the specified execution mode.

Authorization

DBADM authority and authorization to execute the LOAD command are required to execute this function.

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Examples

Example 1: Run the ALTOBJ procedure to alter column CL2 in table T1 from type INTEGER to BIGINT. The original data definition language for table T1 is:

```
CREATE TABLE T1 (CL1 VARCHAR(5), CL2 INTEGER)
```

The ALTOBJ procedure call to alter the column data type is:

```
CALL SYSPROC.ALTOBJ('APPLY_CONTINUE_ON_ERROR',  
  'CREATE TABLE T1 (CL1 VARCHAR(5), CL2 BIGINT)', -1, ?)
```

Note: If you see the following error, try to increase the **applheapsz** parameter value:

SQL0443N Routine "SYSPROC.ALTOBJ" (specific name "ALTOBJ") has returned an error SQLSTATE with diagnostic text "SQL0954 ". SQLSTATE=38553

Example 2: Run the ALTOBJ procedure in VALIDATE mode with *alter-id* input.

```
CALL SYSPROC.ALTOBJ('VALIDATE', CAST (NULL AS VARCHAR(2048)), 123, ?)
```

Usage notes

Although the procedure drops and re-creates the table, the user who created the original table will remain as the table definer. However, an audit will show that the table has been dropped and re-created by the user running the procedure.

This procedure does not support the following alter table operations:

- Altering materialized query tables (MQTs) is not supported. Altering a table which contains an MQT is supported.
- Altering typed tables is not supported.
- Altering a remote table using a nickname is not supported.
- Column sequence cannot be reordered.
- Adding and removing, or renaming and removing columns in one call to the procedure is not supported, but adding and renaming columns is supported. This is because the only way to indicate how the table is to be altered is by the use of the target DDL, rather than column matching information. The following rules are followed by the ALTOBJ procedure when transforming data from the existing table to the altered table:
 1. If the number of columns in the existing table is the same as the altered table, it is assumed that no columns are being added or removed. The columns in this case can only be renamed, and are matched by column index.
 2. If the number of columns in the existing table is less than in the altered table, it is assumed that columns are being added. The columns can be renamed, and the new columns are added at the end. The existing columns are matched by index.
 3. If the number of columns in the existing table is greater than in the altered table, it is assumed that columns are being removed. The columns cannot be renamed and matched by name. The column that is being dropped can be any existing column in the table.
- Structured type UDTs and Reference type UDTs are not supported.
- MQTs defined on a base table which is altered are not populated during the alter table process.

If a table is altered using the ALTOBJ procedure, and the table has an MQT defined, the MQT will be created, but it will not be populated with data.

If a table is altered using the ALTOBJ procedure, and the table has an MQT defined, any columns that are not part of the select result from the table being altered are lost because the MQT content is rebuilt from the new base table.

The definition of the objects might change between ALTOBJ procedure calls because there are no object locks that persist through different sessions.

The table profiles (such as **RUNSTATS** profile) that are associated with the table are lost after going through this extensive alter process.

The SYSTOOLSPACE is used for the routine's operation tables to store metadata; that is, data used to describe database objects and their operation.

If the table has row or column access control activated, ALTOBJ on that table will result in an error (DBA7903).

COMPILATION_ENV table function - Retrieve compilation environment elements

The COMPILATION_ENV table function returns the elements of a compilation environment.

Syntax

►►—COMPILATION_ENV—(—*compilation-env*—)—————►►

The schema is SYSPROC.

Table function parameter

compilation-env

An input argument of type BLOB(2M) that contains a compilation environment obtained from the **comp_env_desc** (compilation environment) monitor element.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

The function returns a table of two columns (see Table 213 on page 1079): NAME VARCHAR(256) and VALUE VARCHAR(1024). The possible values for the compilation environment element names are described in Table 214 on page 1079.

The origin of the element values depends primarily on whether the SQL statement is issued dynamically or bound as part of a package.

The number and types of entries in a compilation environment can change over time as capabilities are added to the database manager. If the compilation environment is from a different database manager level than the level on which this function is executing, only those elements that are recognized by the level of the function are returned. The descriptions of the elements might also vary from release to release.

Examples

Example 1: Request all the elements of a specific compilation environment that was previously captured by a deadlock event monitor. A deadlock event monitor that is created specifying the WITH DETAILS HISTORY option will capture the compilation environment for dynamic SQL statements. This captured environment is what is accepted as input to the table function.

```
SELECT NAME, VALUE
FROM TABLE(SYSPROC.COMPILATION_ENV(:hv1)) AS t
```

```
SELECT NAME, VALUE
FROM TABLE(SYSPROC.COMPIRATION_ENV(:hv1)) AS t
WHERE NAME = 'SCHEMA'
```

1. Obtain the executable ID, which is used to identify the statement of interest, using the following statement:

```
SELECT EXECUTABLE_ID, VARCHAR(STMT_TEXT, 100)
FROM TABLE(MON_GET_PKG_CACHE_STMT(NULL,NULL,NULL,-1)) AS t
```

The following is an example output after executing the preceding statement:

| | |
|---|------------------------------------|
| EXECUTABLE_ID | 2 |
| <hr/> | |
| x'010000000000000001000000000000000000000000020020090914151405241700' | select count(*) from syscat.tables |
| ... | |

[illegible]

The following is an example output after executing the preceding statement:

| 1 | 2 |
|-----------------------|---|
| ISOLATION | CS |
| QUERY_OPTIMIZATION | 5 |
| DEGREE | 1 |
| SQLRULES | DB2 |
| REFRESH_AGE | +000000000000000.000000 |
| RESOLUTION_TIMESTAMP | 2009-09-14-15.14.05.000000 |
| FEDERATED_ASYNCRONY | 0 |
| PATH | "SYSIBM", "SYSFUN", "SYSPROC", "SYSIBMADM", "SWALKTY" |
| MAINTAINED TABLE TYPE | SYSTEM |

9 record(s) selected.

Information returned

Table 213. Information returned by the `COMPILATION_ENV` table function

| Column name | Data type | Description |
|-------------|---------------|---|
| NAME | VARCHAR(256) | Element of compilation environment. See Table 214 for more details. |
| VALUE | VARCHAR(1024) | Value of the element. |

Table 214. Elements of a compilation environment returned by the `COMPILATION_ENV` table function

| Element name | Description |
|--------------|---|
| ISOLATION | The isolation level passed to the SQL compiler. The value is obtained from either the CURRENT ISOLATION special register or the ISOLATION bind option of the current package. |

Table 214. Elements of a compilation environment returned by the `COMPILATION_ENV` table function (continued)

| Element name | Description |
|-----------------------|--|
| QUERY_OPTIMIZATION | The query optimization level passed to the SQL compiler. The value is obtained from either the CURRENT QUERY OPTIMIZATION special register or the QUERYOPT bind option of the current package. |
| DEGREE | The requested degree of intra-parallelism passed to the SQL compiler. The value is obtained from either the CURRENT DEGREE special register or the DEGREE bind option of the current package. |
| SQLRULES | The requested SQL statement behaviors passed to the SQL compiler. The value is derived from the setting of the LANGLVL bind option of the current package. The possible values are 'DB2' or 'SQL92'. |
| REFRESH_AGE | The allowable data latency passed to the SQL compiler. The value is obtained from either the CURRENT REFRESH AGE special register or the REFRESHAGE bind option of the current package. |
| SCHEMA | The default schema passed to the SQL compiler. The value is obtained from either the CURRENT SCHEMA special register or the QUALIFIER bind option of the current package. |
| PATH | The function path passed to the SQL compiler. The value is obtained from either the CURRENT PATH special register or the FUNC_PATH bind option of the current package. |
| TRANSFORM_GROUP | The transform group information passed to the SQL compiler. The value is obtained from either the CURRENT DEFAULT TRANSFORM GROUP special register or the TRANSFORMGROUP package bind option. |
| MAINTAINED_TABLE_TYPE | An indicator of what table types can be considered for optimization, passed to the SQL compiler. The value is obtained from the CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register. |
| RESOLUTION_TIMESTAMP | The timestamp that is to be used by the SQL compiler for resolving items such as function and data type references in an SQL statement. This timestamp is either the current timestamp or the timestamp of the last explicit bind operation for the current package. |
| FEDERATED_ASYNCHRONY | The requested degree of federated asynchrony parallelism passed to the SQL compiler. The value is obtained from either the CURRENT FEDERATED ASYNCHRONY special register or the FEDERATED_ASYNCHRONY bind option of the current package. |

CONTACTGROUPS administrative view - Retrieve the list of contact groups

The CONTACTGROUPS administrative view returns the list of contact groups, which can be defined locally on the system or in a global list.

The setting of the Database Administration Server (DAS) **contact_host** configuration parameter determines whether the list is local or global.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the CONTACTGROUPS administrative view
- CONTROL privilege on the CONTACTGROUPS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve all contact group lists.

```
SELECT * FROM SYSIBMADM.CONTACTGROUPS
```

The following is an example of output for this query.

| NAME | DESCRIPTION | MEMBERNAME | MEMBERTYPE |
|--------|-------------------------|------------|------------|
| group1 | DBA Group1 Contact List | name1 | CONTACT |
| group1 | DBA Group1 Contact List | name9 | CONTACT |
| group2 | DBA Group2 List | name2 | CONTACT |
| group3 | | group2 | GROUP |
| group5 | DBA Group5 | group2 | GROUP |
| group6 | DBA Group6 | group3 | GROUP |
| group7 | | name1 | CONTACT |

7 record(s) selected.

Usage note

The DAS must have been created and be running.

Information returned

Table 215. Information returned by the CONTACTGROUPS administrative view

| Column name | Data type | Description |
|-------------|--------------|---|
| NAME | VARCHAR(128) | Name of the contact group. |
| DESCRIPTION | VARCHAR(128) | Description of the contact group. |
| MEMBERNAME | VARCHAR(128) | Name of the member in the contact group. This name can refer to a contact or another contact group. |
| MEMBERTYPE | VARCHAR(7) | Type of member in the contact group. The type is either CONTACT or GROUP. |

CONTACTS administrative view - Retrieve list of contacts

The CONTACTS administrative view returns the list of contacts defined on the database server.

The setting of the Database Administration Server (DAS) **contact_host** configuration parameter determines whether the list is local or global.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the CONTACTS administrative view
- CONTROL privilege on the CONTACTS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve all contacts.

```
SELECT * FROM SYSIBMADM.CONTACTS
```

The following is an example of output for this query.

| NAME | TYPE | ADDRESS | MAX_PAGE_LENGTH | DESCRIPTION |
|-------|-------|------------------|-----------------|----------------------|
| user1 | EMAIL | user3@ca.ibm.com | | - DBA Extraordinaire |
| user2 | EMAIL | user2@ca.ibm.com | | - DBA on Email |
| user3 | PAGE | user3@ca.ibm.com | 128 | DBA on Page |
| user5 | EMAIL | user2@ca.ibm.com | | - DBA Extraordinaire |

4 record(s) selected.

Usage note

The DAS must have been created and be running.

Information returned

Table 216. Information returned by the CONTACTS administrative view

| Column name | Data type | Description |
|-----------------|--------------|---|
| NAME | VARCHAR(128) | Name of contact. |
| TYPE | VARCHAR(5) | Type of contact: <ul style="list-style-type: none">• 'EMAIL'• 'PAGE' |
| ADDRESS | VARCHAR(128) | address - IP address from which the connection was initiated |
| MAX_PAGE_LENGTH | INTEGER | Maximum message length. Used for example, if the paging service has a message-length restriction. |
| DESCRIPTION | VARCHAR(128) | Description of contact. |

DB_HISTORY administrative view - Retrieve history file information

The DB_HISTORY administrative view returns information from the history files from all database partitions.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the DB_HISTORY administrative view
- CONTROL privilege on the DB_HISTORY administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Usage note

When a data partitioned table is reorganized, one record for each reorganized data partition is returned. If only a specific data partition of a data partitioned table is reorganized, only a record the for the partition is returned.

Example

Select the database partition number, entry ID, operation, start time, and status information from the database history files for all the database partitions of the database to which the client is currently connected.

```
SELECT DBPARTITIONNUM, EID, OPERATION, START_TIME, ENTRY_STATUS
FROM SYSIBMADM.DB_HISTORY
```

The following is an example of output for this query.

| DBPARTITIONNUM | EID | OPERATION | START_TIME | ENTRY_STATUS |
|----------------|-----|-----------|----------------|--------------|
| 0 | 1 | A | 20051109185510 | A |

1 record(s) selected.

Information returned

Table 217. Information returned by the DB_HISTORY administrative view

| Column name | Data type | Description |
|----------------|-------------|---|
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| EID | BIGINT | Number that uniquely identifies an entry in the history file. |
| START_TIME | VARCHAR(14) | start_time - Event Start Time monitor element |
| SEQNUM | SMALLINT | Positive integer which identifies the various parts of an image, such as sequences or sessions. |
| END_TIME | VARCHAR(14) | Timestamp marking the end of a logged event. |

Table 217. Information returned by the DB_HISTORY administrative view (continued)

| Column name | Data type | Description |
|---------------|---------------|--|
| NUM_LOG_ELEMS | INTEGER | Number of log streams that will be returned in the lists of log stream data in the FIRSTLOG and LASTLOG columns. |
| FIRSTLOG | VARCHAR(4000) | Name of the earliest transaction log associated with an event. <ul style="list-style-type: none"> In Db2 pureScale environments, where one or more log streams can exist, the entry uses the following format:
(<log_stream_id>: <log_file_name> , <log_stream_id>: <log_file_name> , ...) In non-Db2 pureScale environments, where only a single log stream exists per database partition, the entry uses the following format:
(<log_file_name>) |
| LASTLOG | VARCHAR(4000) | Name of the latest transaction log associated with an event. <ul style="list-style-type: none"> In Db2 pureScale environments, where one or more log streams can exist, the entry uses the following format:
(<log_stream_id>: <log_file_name> , <log_stream_id>: <log_file_name> , ...) In non-Db2 pureScale environments, where only a single log stream exists per database partition, the entry uses the following format:
(<log_file_name>) |
| BACKUP_ID | VARCHAR(24) | Backup identifier or unique table identifier. |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name monitor element |
| TABNAME | VARCHAR(128) | table_name - Table name monitor element |
| COMMENT | VARCHAR(254) | System-generated comment text associated with a logged event.
Note: Only the first 30 characters are stored in the history file |
| CMD_TEXT | CLOB(2 M) | Data definition language associated with a logged event. |
| NUM_TBSPS | INTEGER | num_tbsps - Number of table spaces monitor element |
| TBSPNAMES | CLOB(5 M) | Names of the table spaces associated with a logged event. |
| OPERATION | CHAR(1) | Operation identifier. See Table 218 on page 1086 for possible values. |
| OPERATIONTYPE | CHAR(1) | Action identifier for an operation. See Table 218 on page 1086 for possible values. |
| OBJECTTYPE | CHAR(1) | Identifier for the target object of an operation. The possible values are: D for full database, I for index, P for table space, R for partitioned table, and T for table. |
| LOCATION | VARCHAR(255) | Full path name for files, such as backup images or load input files, that are associated with logged events. In the case of CLI LOAD or LOAD from CURSOR, the location is an SQL statement. |

Table 217. Information returned by the DB_HISTORY administrative view (continued)

| Column name | Data type | Description |
|--------------|-------------|--|
| DEVICETYPE | CHAR(1) | Identifier for the device type associated with a logged event. This field determines how the LOCATION field is interpreted: <ul style="list-style-type: none"> • A for TSM • C for client • D for disk • F for snapshot backup • K for diskette • L for local • N (generated internally by the database system) • O for other (for other vendor device support) • P for pipe • Q for cursor • R for remote fetch data • S for server • T for tape • U for user exit • X for X/Open XBSA interface |
| ENTRY_STATUS | CHAR(1) | Identifier for the status of an entry in the history file: <ul style="list-style-type: none"> • A for active • D for deleted (future use) • E for expired, I for inactive • N for not yet committed • Y for committed or active |
| SQLCAID | VARCHAR(8) | An "eye catcher" for storage dumps containing 'SQLCA', as it appears in the SQLCAID field of the SQL communications area (SQLCA). |
| SQLCABC | INTEGER | Length of the SQLCA, as it appears in the SQLCABC field of the SQLCA. |
| SQLCODE | INTEGER | SQL return code, as it appears in the SQLCODE field of the SQLCA. |
| SQLERRML | SMALLINT | Length indicator for SQLERRMC, as it appears in the SQLERRML field of the SQLCA. |
| SQLERRMC | VARCHAR(70) | Contains one or more tokens, separated by X'FF', as they appear in the SQLERRMC field of the SQLCA. These tokens are substituted for variables in the descriptions of error conditions. |
| SQLERRP | VARCHAR(8) | A three-letter identifier indicating the product, followed by five alphanumeric characters indicating the version, release, and modification level of the product, as they appear in the SQLERRP field of the SQLCA. |
| SQLERRD1 | INTEGER | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> . |
| SQLERRD2 | INTEGER | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> . |
| SQLERRD3 | INTEGER | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> . |
| SQLERRD4 | INTEGER | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> . |
| SQLERRD5 | INTEGER | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> . |
| SQLERRD6 | INTEGER | See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> . |

Table 217. Information returned by the DB_HISTORY administrative view (continued)

| Column name | Data type | Description |
|-------------|-------------|---|
| SQLWARN | VARCHAR(11) | A set of warning indicators, each containing a blank or 'W'. See "SQLCA (SQL communications area)" in <i>SQL Reference Volume 1</i> . |
| SQLSTATE | VARCHAR(5) | A return code that indicates the outcome of the most recently executed SQL statement, as it appears in the SQLSTATE field of the SQLCA. |

Table 218. OPERATION and OPERATIONTYPE values

| Operation value | Operation value description | Operation type |
|-----------------|-----------------------------|--|
| A | Add table space | None |
| B | Backup | Operation types are: <ul style="list-style-type: none"> • D = delta offline • E = delta online • F = offline • I = incremental offline • N = online • O = incremental online |
| C | Load copy | None |
| D | Dropped table | None |
| F | Rollforward | Operation types are: <ul style="list-style-type: none"> • E = end of logs • P = point in time |
| G | Reorganize table | Operation types are: <ul style="list-style-type: none"> • F = offline • N = online |
| L | Load | Operation types are: <ul style="list-style-type: none"> • I = insert • R = replace |
| N | Rename table space | None |
| O | Drop table space | None |
| Q | Quiesce | Operation types are: <ul style="list-style-type: none"> • S = quiesce share • U = quiesce update • X = quiesce exclusive • Z = quiesce reset |
| R | Restore | Operation types are: <ul style="list-style-type: none"> • F = offline • I = incremental offline • N = online • O = incremental online • R = rebuild |
| T | Alter table space | Operation types are: <ul style="list-style-type: none"> • C = add containers • R = rebalance |
| U | Unload | None |

Table 218. OPERATION and OPERATIONTYPE values (continued)

| Operation value | Operation value description | Operation type |
|-----------------|-----------------------------|---|
| X | Archive logs | Operation types are: <ul style="list-style-type: none"> • F = fail archive path • M = mirror log path • N = forced truncation via ARCHIVE LOG command • P = primary log path • 1 = first log archive method • 2 = second log archive method |

DBPATHS administrative view and ADMIN_LIST_DB_PATHS table function - Retrieve database paths

The DBPATHS administrative view and the ADMIN_LIST_DB_PATHS table function return the values for database paths that are required for tasks such as creating split mirror backups.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “DBPATHS administrative view”
- “ADMIN_LIST_DB_PATHS table function” on page 1088

DBPATHS administrative view

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the DBPATHS administrative view
- CONTROL privilege on the DBPATHS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

The following statement retrieves all database paths:

```
select dbpartitionnum, substr(type,1,20) as type, path from sysibmadm.dbpaths
```

The following is an example of output for this query.

| DBPARTITIONNUM | TYPE | PATH |
|----------------|--------------------|--|
| 0 | LOGPATH | /home/sun/sun/NODE0000/SQL00001/LOGSTREAM0000/ |
| 0 | DB_STORAGE_PATH | /home/sun/ |
| 0 | LOCAL_DB_DIRECTORY | /home/sun/sun/NODE0000/sqlbdir/ |
| 0 | DBPATH | /home/sun/sun/NODE0000/SQL00001/ |
| 0 | DBPATH | /home/sun/sun/NODE0000/SQL00001/MEMBER0000/ |

5 record(s) selected.

ADMIN_LIST_DB_PATHS table function

The ADMIN_LIST_DB_PATHS table function returns the list of files required for backup mechanisms such as split mirror backup.

Syntax

►►—ADMIN_LIST_DB_PATHS—(—)—————►►

The schema is SYSPROC.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the ADMIN_LIST_DB_PATHS table function
- DATAACCESS authority

In addition, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMOINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

The ADMIN_LIST_DB_PATHS table function can be invoked as follows:

```
SELECT DBPARTITIONNUM, TYPE, PATH FROM TABLE(ADMIN_LIST_DB_PATHS()) AS FILES
```

The following is an example of output from this query.

| DBPARTITIONNUM | TYPE |
|----------------|--------------------|
| 2 | LOGPATH |
| 2 | DB_STORAGE_PATH |
| 2 | TBSP_DIRECTORY |
| 2 | TBSP_CONTAINER |
| 2 | LOCAL_DB_DIRECTORY |
| 2 | DBPATH |
| 2 | DBPATH |
| 0 | LOGPATH |
| 0 | DB_STORAGE_PATH |

```

0 TBSP_DIRECTORY
0 TBSP_CONTAINER
0 LOCAL_DB_DIRECTORY
0 DBPATH
0 DBPATH
1 LOGPATH
1 DB_STORAGE_PATH
1 TBSP_DIRECTORY
1 TBSP_CONTAINER
1 LOCAL_DB_DIRECTORY
1 DBPATH
1 DBPATH

```

PATH

```

-----
/home/sun/sun/NODE0002/SQL00001/LOGSTREAM0002/
/home/sun/
/home/sun/tablespace/sms/sms2/
/home/sun/tablespace/dms/dms2
/home/sun/sun/NODE0002/sqlbdir/
/home/sun/sun/NODE0002/SQL00001/
/home/sun/sun/NODE0002/SQL00001/MEMBER0002/
/home/sun/sun/NODE0000/SQL00001/LOGSTREAM0000/
/home/sun/
/home/sun/tablespace/sms/sms0/
/home/sun/tablespace/dms/dms0
/home/sun/sun/NODE0000/sqlbdir/
/home/sun/sun/NODE0000/SQL00001/
/home/sun/sun/NODE0000/SQL00001/MEMBER0000/
/home/sun/sun/NODE0001/SQL00001/LOGSTREAM0001/
/home/sun/
/home/sun/tablespace/sms/sms1/
/home/sun/tablespace/dms/dms1
/home/sun/sun/NODE0001/sqlbdir/
/home/sun/sun/NODE0001/SQL00001/
/home/sun/sun/NODE0001/SQL00001/MEMBER0001/

```

21 record(s) selected.

If the storage library performing the split mirror operation treats files and directories on raw devices differently than those on regular file systems, you can use the following query to obtain the list for all locations on raw devices:

```

SELECT DBPARTITIONNUM, TYPE, PATH
FROM TABLE(ADMIN_LIST_DB_PATHS()) AS FILES
WHERE TYPE LIKE '%_DEVICE%'

```

Second, the list of files and directories on regular file systems:

```

SELECT DBPARTITIONNUM, TYPE, PATH
FROM TABLE(ADMIN_LIST_DB_PATHS()) AS FILES
WHERE TYPE NOT LIKE '%_DEVICE%'

```

Information returned

Table 219. Information returned by the DBPATHS administrative view and the ADMIN_LIST_DB_PATHS table function

| Column name | Data type | Description |
|----------------|-----------|--|
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

Table 219. Information returned by the DBPATHS administrative view and the ADMIN_LIST_DB_PATHS table function (continued)

| Column name | Data type | Description |
|-------------|---------------|---|
| TYPE | VARCHAR(64) | Describes the type of database object that the path belongs to. For example the path to the log directory indicated by the LOGPATH database configuration parameter would be shown in this column as LOGPATH. See Table 220 for a list of possible return values. |
| PATH | VARCHAR(5000) | Path to location where the database manager has a file or directory located. If the path ends with the file system delimiter ('/' on UNIX environments, '\' on Windows environments), the path points to a directory. |

Table 220. TYPE column values

| Type value | Description |
|--------------------|--|
| TBSP_DEVICE | Raw device for a database managed space (DMS) table space. |
| TBSP_CONTAINER | File container for a DMS table space. |
| TBSP_DIRECTORY | Directory for a system managed space (SMS) table space. |
| LOGPATH | Primary log path. |
| MIRRORLOGPATH | Database configuration mirror log path. |
| DB_STORAGE_PATH | Automatic storage path. |
| DBPATH | Database directory path. |
| LOCAL_DB_DIRECTORY | Path to the local database directory. |

- For table spaces using automatic storage, both used and unused storage paths are returned. You require the unused automatic storage paths if you restore by using a split mirror backup.
Consider the following example. A split mirror backup is taken on a production system. After the backup is completed, the automatic storage paths that were not in use before the backup are now in use in production. Assume that there is now a need to restore the database by using the split mirror backup. At this point, it is necessary to roll forward the logs from the production database. To roll forward the logs, all of the automatic storage paths are required, because all automatic storage paths are now in use.
- Table space containers that are managed by automatic storage are not returned individually. Instead, they are reflected in the automatic storage path column.
- The automatic storage paths are returned once per database partition.
- The values returned for the **logpath** and **mirrorlogpath** configuration parameters are the values stored in memory. Changed values that are stored on disk, which are applicable only after a database restart, are not returned.

- If you use output from the `SELECT * FROM SYSIBMADM.DBPATHS` query to create a **db2relocatedb** command configuration file (a file containing the configuration information that is necessary for relocating a database), you must modify the DBPATH output appropriately before you can use it in the configuration file.

For example, consider the following DBPATH output:

```
/storage/svtdbm3/svtdbm3/NODE0000/SQL00001/
```

You can use this output to specify the value of the DB_PATH parameter in a **db2relocatedb** command configuration file, as follows:

```
DB_PATH=/storage/svtdbm3,/storage_copy2/svtdbm3
```

- The LOCAL_DB_DIRECTORY path might contain information belonging to multiple databases. Because each database that you create in the same directory does not have its own sqlbdir file, ensure that the target system to which you copy files does not have any databases already in that path.
- If two or more databases share at least one automatic storage path, the split mirror operation for one of these databases might affect more than one database, causing I/O problems for the databases that you did not intend to split.
- The DB_STORAGE_PATH type includes all storage paths from all defined storage groups. If a storage path is used by multiple storage groups or is specified multiple times in the same storage group, one record is returned for each occurrence of the path in a storage group.

Restriction

You cannot call the administrative view when the database is in WRITE SUSPEND mode. Also, you must ensure that the physical layout of the database does not change between the invocation of the view and the activation of WRITE SUSPEND mode, which is needed to perform a split mirror operation. You might not be able to restore from the split mirror backup image if, for example, the table space layout changed in that time.

GET_DBSIZE_INFO

The GET_DBSIZE_INFO procedure calculates the database size and maximum capacity.

Authorization

To execute the procedure, SYSMON authority is required, in addition to one of the following authorizations:

- DATAACCESS authority
- DBADM authority
- SQLADM authority
- EXECUTE privilege on the GET_DBSIZE_INFO, WLM_SET_CONN_ENV, MON_GET_CONTAINER, and MON_GET_TABLESPACE routines

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Syntax

```

▶▶—GET_DBSIZE_INFO—(—snapshot-timestamp—,—dbsize—,—dbcapacity—,——————▶
▶—refresh-window—)—————▶▶

```

The schema is SYSPROC.

Procedure parameters

snapshot-timestamp

An output parameter of type `TIMESTAMP` that returns the time at which *dbsize* and *dbcapacity* were calculated. This timestamp, along with the value of *refresh-window*, is used to determine when the cached values in the SYSTOOLS.STMG_DBSIZE_INFO table need to be refreshed.

dbsize

An output parameter of type `BIGINT` that returns the size of the database (in bytes). The database size is calculated as follows: *dbsize* = sum (used_pages * page_size) for each table space (SMS & DMS).

dbcapacity

An output parameter of type `BIGINT` that returns the database capacity (in bytes). This value is not available on partitioned database systems. The database capacity is calculated as follows: *dbcapacity* = SUM (DMS usable_pages * page size) + SUM (SMS container size + file system free size per container). If multiple SMS containers are defined on the same file system, the file system free size is included only once in the calculation of capacity.

refresh-window

An input argument of type `INTEGER` that specifies the number of minutes until the cached values for database size and capacity are to be refreshed. Specify -1 for the default refresh window of 30 minutes. A refresh window of 0 forces an immediate refreshing of the cached values.

Usage notes

The calculated values are returned as procedure output parameters and are cached in the SYSTOOLS.STMG_DBSIZE_INFO table. The procedure caches these values because the calculations are costly. The SYSTOOLS.STMG_DBSIZE_INFO table is created automatically the first time the procedure executes. If there are values cached in the SYSTOOLS.STMG_DBSIZE_INFO table and they are current enough, as determined by the *snapshot-timestamp* and *refresh-window* values, these cached values are returned. If the cached values are not current enough, new cached values are calculated, inserted into the SYSTOOLS.STMG_DBSIZE_INFO table and returned, and the *snapshot-timestamp* value is updated.

To ensure that the data is returned by all partitions for a global table space snapshot, the database must be activated.

The SYSTOOLSPACE is used for the routine's operation tables to store metadata; that is, data used to describe database objects and their operation.

Examples

Example 1: Get the database size and capacity using a default refresh window of 30 minutes. The database size and capacity will be recalculated when the cached data is older than 30 minutes.

```
CALL GET_DBSIZE_INFO(?, ?, ?, -1)
```

The procedure returns:

Value of output parameters

Parameter Name : SNAPSHOTTIMESTAMP

Parameter Value : 2004-02-29-18.31.55.178000

Parameter Name : DATABASESIZE

Parameter Value : 22302720

Parameter Name : DATABASECAPACITY

Parameter Value : 4684793856

Return Status = 0

Example 2: Get the database size and capacity using a refresh window of 0 minutes. The database size and capacity will be recalculated immediately.

CALL GET_DBSIZE_INFO(?, ?, ?, 0)

The procedure returns:

Value of output parameters

Parameter Name : SNAPSHOTTIMESTAMP

Parameter Value : 2004-02-29-18.33.34.561000

Parameter Name : DATABASESIZE

Parameter Value : 22302720

Parameter Name : DATABASECAPACITY

Parameter Value : 4684859392

Return Status = 0

Example 3: Get the database size and capacity using a refresh window of 24 hours. The database size and capacity will be recalculated when the cached data is older than 1440 minutes.

CALL GET_DBSIZE_INFO(?, ?, ?, 1440)

The procedure returns:

Value of output parameters

Parameter Name : SNAPSHOTTIMESTAMP

Parameter Value : 2004-02-29-18.33.34.561000

Parameter Name : DATABASESIZE

Parameter Value : 22302720

Parameter Name : DATABASECAPACITY

Parameter Value : 4684859392

Return Status = 0

NOTIFICATIONLIST administrative view - Retrieve contact list for health notification

The NOTIFICATIONLIST administrative view returns the list of contacts and contact groups that are notified about the health of an instance.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the NOTIFICATIONLIST administrative view
- CONTROL privilege on the NOTIFICATIONLIST administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

The NOTIFICATIONLIST administrative view returns the list of contacts and contact groups that are notified about the health of an instance.

The following is an example of output for this query.

| NAME | TYPE |
|--------|---------|
| group3 | GROUP |
| user4 | CONTACT |
| group3 | GROUP |

3 record(s) selected.

Information returned

Table 221. Information returned by the NOTIFICATIONLIST administrative view

| Column name | Data type | Description |
|-------------|--------------|--|
| NAME | VARCHAR(128) | Name of contact. |
| TYPE | VARCHAR(7) | Type of contact: <ul style="list-style-type: none">• 'CONTACT'• 'GROUP' |

PD_GET_DIAG_HIST - Return records from a given facility

The PD_GET_DIAG_HIST table function returns log records, event records and notification records from a given facility.

Options are also provided to filter based on the type of record, customer impact value of the record, and from-until timestamps.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Syntax

```
►►PD_GET_DIAG_HIST(—facility—,—rectype—,—impact—,—start_time—,—end_time—,—member—)————►◄
```

The schema is SYSPROC.

Table function parameters

facility

An optional input argument of type VARCHAR(20) that specifies the facility from which records are to be returned. A facility is a logical grouping that records relate to. The possible values are:

- ALL: Returns records from all facilities
- MAIN: Returns records from the general diagnostic logs. This currently means the **db2diag** log files, the admin notification log, and the rotating event logs.
- OPTSTATS: Return records related to optimizer statistics

If this parameter is null or an empty string ("), 'ALL' is the default.

rectype

An optional input argument of type VARCHAR(30) that specifies which record type to return. A combination of types separated by '+' are supported, for example: 'D + EI'. The possible values are:

- 'ALL': Return all record types.
- 'D': Return all diagnostic records.
- 'E': Return all event records.
- 'DI': Internal diagnostic records. These are non-translated diagnostic record that are used by IBM support in a diagnostic situation.
- 'DX': External diagnostic records. These are translated diagnostic that are of use to the user. These records are the notification records.
- 'EI': Internal event record. These are event record that are used by IBM support in a diagnostic situation.
- 'EX': External event record. These are diagnostic record that are of use to the user.

If this parameter is null or an empty string ("), all records are returned.

impact

An optional input argument of type VARCHAR(18) that specifies the minimum customer impact level of the record returned. The possible values are:

- 'NONE'
- 'UNLIKELY'
- 'POTENTIAL'
- 'IMMEDIATE'
- 'CRITICAL'

If this parameter is null or an empty string ("), all records are returned.

start_time

An optional input argument of type `TIMESTAMP` that specifies a valid timestamp. Entries are returned if their timestamp is more recent than this value. If this parameter is null, records are returned regardless of how old they are.

end_time

An optional input argument of type `TIMESTAMP` that specifies a valid timestamp. Entries are returned if their timestamp is older than this value. If this parameter is null, records are returned regardless of how recent they are.

member

An optional input argument of type `INTEGER` that specifies a valid database member from which the records should be fetched. Specify -1 or null for the current member, or -2 for information from all active database members. An active database member is where the database is available for connection and use by applications. If the parameter is not specified, the default value is all active database members.

Information returned

Table 222. Information returned by the `PD_GET_DIAG_HIST` table function

| Column Name | Data Type | Description |
|----------------|--------------|--|
| FACILITY | VARCHAR(20) | A facility is a logical grouping which records relate to. The possible values are: <ul style="list-style-type: none">• ALL: Returns records from all facilities• MAIN: Returns records from the general diagnostic logs. This currently means the db2diag log files, the admin notification log, and the rotating event logs.• OPTSTATS: Return records related to optimizer statistics |
| RECTYPE | VARCHAR(3) | The type of record. The possible values are: <ul style="list-style-type: none">• 'DI': Internal diagnostic record• 'DX': External diagnostic record• 'EI': Internal event record• 'EX': External event record |
| TIMESTAMP | TIMESTAMP | The time that the message was created. |
| TIMEZONE | INTEGER | The time difference (in minutes) from the Universal Coordinated Time (UCT). For example, -300 is EST. |
| INSTANCENAME | VARCHAR(128) | The name of the instance where the message was created. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| LEVEL | CHAR(1) | The severity level of the record. The possible values are: <ul style="list-style-type: none">• 'C': Critical• 'E': Error• 'I': Informational• 'S': Severe• 'W': Warning |

Table 222. Information returned by the PD_GET_DIAG_HIST table function (continued)

| Column Name | Data Type | Description |
|-----------------|--------------|--|
| IMPACT | VARCHAR(18) | Qualifies the impact of this message from a user's perspective. This clarifies the impact of the message on the business process the database system is part of. Possible values are: <ul style="list-style-type: none"> • 'CRITICAL' • 'IMMEDIATE' • 'NONE' • 'POTENTIAL' • 'UNLIKELY' |
| DBNAME | VARCHAR(128) | The name of the database being accessed while this message was created. |
| EDU_ID | BIGINT | edu_ID - Engine dispatchable unit ID monitor element |
| EDUNAME | VARCHAR(64) | The name of the engine Dispatched Unit that created this message. |
| PID | BIGINT | The operating system process identifier that created this message. |
| PROCESS_NAME | VARCHAR(255) | The operating system process name that created this message. |
| TID | BIGINT | The thread numeric identifier that created this message. |
| APPLNAME | VARCHAR(255) | The name of the client application that initiated the connection, if it is available. |
| APPL_ID | VARCHAR(64) | appl_id - Application ID monitor element |
| APPLHANDLE | VARCHAR(9) | A system-wide unique identifier for the application that initiated the connection when available. This is synonymous to agent ID. The identifier consists of the coordinating member number and a 16-bit counter separated by a '-'. The format is as follows: 'nnn-xxxxx' |
| AUTH_ID | VARCHAR(30) | auth_id - Authorization ID monitor element |
| PRODUCT | VARCHAR(50) | The name of the product that created the message. For example 'Db2 Common'. |
| COMPONENT | VARCHAR(255) | The name of the component that created the message. |
| FUNCTION | VARCHAR(255) | The name of the function that generated the message. |
| PROBE | INTEGER | Probe point number used to identify where the message was generated in the function. |
| CALLEDPRODUCT | VARCHAR(50) | The name of the product at the source of the error. This is used when the source of an error is not where the message was created. |
| CALLEDCOMPONENT | VARCHAR(255) | The name of the component at the source of the error. This is used when the source of an error is not where the message was created. |
| CALLEDFUNCTION | VARCHAR(255) | The name of the function at the source of the error. This is used when the source of an error is not where the message was created. |
| OSERR | INTEGER | The operating system error number. |
| RETCODE | INTEGER | The product specific return code. |

Table 222. Information returned by the PD_GET_DIAG_HIST table function (continued)

| Column Name | Data Type | Description |
|-------------------|--------------|--|
| MSGNUM | INTEGER | The numeric message number for the associated message, if it is available. For example, this is the numeric portion of ADM7513W. |
| MSGTYPE | CHAR(3) | The type related to the message identifier, if it is available. For example, ADM is used for administration notification log messages. |
| MSG | CLOB(16K) | The short description text for this record. This is the translated message text corresponding to the MSGNUM, and MSGTYPE for translated messages. For non-translated messages, this is the short description. For example : 'Bringing down all db2fmp processes as part of db2stop'. |
| OBJTYPE | VARCHAR(64) | Type of object. One of: <ul style="list-style-type: none"> • EVALUATION • INDEX STATS • INITIAL SAMPLING • TABLE STATS • TABLE AND INDEX STATS • SAMPLING TEST • STATS DAEMON |
| OBJNAME | VARCHAR(255) | The name of the object the event relates to, if it is available. |
| OBJNAME_QUALIFIER | VARCHAR(255) | Additional information about the object, if it is available. |

Table 222. Information returned by the PD_GET_DIAG_HIST table function (continued)

| Column Name | Data Type | Description |
|-------------|--------------|---|
| EVENTTYPE | VARCHAR(24) | <p>The event type is the action or verb associated with this event. The possible values are:</p> <ul style="list-style-type: none"> • 'ACCEPT' • 'ACCESS' • 'ADD' • 'ALTER' • 'ASSOCIATE' • 'AVAILABLE' • 'BRINGDOWN' • 'CHANGE' • 'CHANGECFG' • 'CLOSE' • 'COLLECT' • 'CONNECT' • 'CREATE' • 'DEPENDENCY' • 'DESTROY' • 'DISASSOCIATE' • 'DISCONNECT' • 'DISPATCH' • 'DROP' • 'FINI' • 'FREE' • 'GET' • 'INIT' • 'INTERRUPT' • 'OPEN','READ' • 'RECV' • 'REPLY' • 'REPORT' • 'REQUEST' • 'RESET' • 'SEND' • 'START' • 'STARTUP' • 'STOP' • 'SWITCH' • 'TERMINATE' • 'TRANSFER' • 'WAIT' • 'WORK' • 'WRITE' |
| EVENTDESC | VARCHAR(256) | A short representation of the key fields for this event. |

Table 222. Information returned by the PD_GET_DIAG_HIST table function (continued)

| Column Name | Data Type | Description |
|---------------------------|--------------|--|
| FIRST_EVENTQUALIFIERTYPE | VARCHAR(64) | The type of the first event qualifier. Event qualifiers are used to describe what was affected by the event. The possible values are: <ul style="list-style-type: none"> • 'AT' • 'BY' • 'CONTEXT' • 'DUE TO' • 'FOR' • 'FROM' • 'ON' • 'TO' If <i>facility</i> is OPTSTATS, the only value is 'AT'. |
| FIRST_EVENTQUALIFIER | CLOB(16K) | The first qualifier for the event. If <i>facility</i> is OPTSTATS, this will be a timestamp indicating when the statistics collection occurred. |
| SECOND_EVENTQUALIFIERTYPE | VARCHAR(64) | The type of the second event qualifier. If <i>facility</i> is OPTSTATS, the value is 'BY'. |
| SECOND_EVENTQUALIFIER | CLOB(16K) | The second qualifier for the event. If <i>facility</i> is OPTSTATS, the possible values are: <ul style="list-style-type: none"> • Asynchronous • FABRICATE • FABRICATE PARTIAL • SYNCHRONOUS • SYNCHRONOUS SAMPLED • USER |
| THIRD_EVENTQUALIFIERTYPE | VARCHAR(64) | The type of the third event qualifier. If <i>facility</i> is OPTSTATS, the value is 'DUE TO'. |
| THIRD_EVENTQUALIFIER | CLOB(16K) | The third qualifier for the event. If <i>facility</i> is OPTSTATS, the possible values are: <ul style="list-style-type: none"> • Conflict • Error • Object unavailable • RUNSTATS error • Timeout |
| EVENTSTATE | VARCHAR(255) | State of the object or action as a result of the event. This can also contain a percentage indicating the progression of the event. |

Table 222. Information returned by the PD_GET_DIAG_HIST table function (continued)

| Column Name | Data Type | Description |
|----------------|--------------|--|
| EVENTATTRIBUTE | VARCHAR(255) | The event attributes. This is a list of attributes associated with the event. when more than one attribute is used, the list is separated by '+' characters. For example 'CACHED + LOGICAL + AUTO'. The possible values are: <ul style="list-style-type: none"> • 'ASYNCH' • 'AUTO' • 'CACHED' • 'DIRECT' • 'EXTERNAL' • 'INDIRECT' • 'INTERNAL' • 'LOGICAL' • 'PERMANENT' • 'PHYSICAL' • 'SYNC' • 'TEMPORARY' |
| EVENTSTACK | CLOB(16K) | The logical event stack at the point the record was logged when applicable. |
| CALLSTACK | CLOB(16K) | The operating system stack dump for the thread that generated this record when applicable. |
| DUMPFIL | CLOB(5000) | The name of the secondary dump file associated with the log record when applicable. This is a fully qualified path to a file or directory where additional information related to the message can be retrieved. |
| FULLREC | CLOB(16K) | Formatted text version of the entire record. This section also contains additional DATA fields. |
| MEMBER | SMALLINT | member - Database member monitor element |
| HOSTNAME | VARCHAR(255) | hostname - Host name |

Usage note

The PD_GET_DIAG_HIST table function requires that the associated database has a temporary table space with minimum page size of 8K. If the page size is less than 8K, the function will return an SQL1585N error message.

Examples

Example 1: Retrieve records from a given facility.

```
SELECT FACILITY, RECTYPE, TIMESTAMP, IMPACT, SUBSTR(MSG,1, 50) AS MSG
FROM TABLE (PD_GET_DIAG_HIST( 'MAIN', 'E', '', NULL,
NULL ) ) AS T
WHERE T.PROCESS_NAME = 'db2star2' OR T.PROCESS_NAME = 'db2stop2'
```

The following is an example of output from this query.

| FACILITY | RECTYPE | TIMESTAMP | ... |
|----------|---------|----------------------------|-----|
| ----- | ----- | ----- | ... |
| MAIN | EX | 2007-06-25-11.34.05.756171 | ... |
| MAIN | EX | 2007-06-25-11.34.25.946646 | ... |

2 record(s) selected.

Output from this query (continued).

| ... | IMPACT | MSG |
|-------|--------|--|
| ----- | ----- | ----- |
| ... | - | ADM7514W Database manager has stopped. |
| ... | - | ADM7513W Database manager has started. |

Example 2: Retrieve records for a specific member.

```
SELECT MEMBER,DBPARTITIONNUM, FACILITY, RECTYPE, TIMESTAMP, IMPACT,
SUBSTR(MSG,1, 50) AS MSG FROM TABLE (PD_GET_DIAG_HIST('MAIN', 'E', '',
CAST (NULL AS TIMESTAMP), CAST (NULL AS TIMESTAMP), NULL ) ) AS T
WHERE T.PROCESS_NAME = 'db2star2' OR T.PROCESS_NAME = 'db2stop2' ORDER BY MEMBER
```

The following is an example of output from this query.

| MEMBER | DBPARTITIONNUM | FACILITY | RECTYPE | TIMESTAMP | ... |
|--------|----------------|----------|---------|----------------------------|-----|
| ----- | ----- | ----- | ----- | ----- | ... |
| 0 | 0 | MAIN | EI | 2011-04-28-09.44.57.720041 | ... |
| 0 | 0 | MAIN | EI | 2011-04-28-09.44.57.723015 | ... |
| 0 | 0 | MAIN | EI | 2011-04-28-09.44.57.723736 | ... |
| 0 | 0 | MAIN | EI | 2011-04-28-09.44.59.409586 | ... |
| 0 | 0 | MAIN | EX | 2011-04-28-09.45.01.554096 | ... |
| 0 | 0 | MAIN | EI | 2011-04-28-09.45.01.605231 | ... |
| 0 | 0 | MAIN | EI | 2011-04-28-12.34.20.571551 | ... |
| 0 | 0 | MAIN | EI | 2011-04-28-12.34.20.574612 | ... |
| 0 | 0 | MAIN | EI | 2011-04-28-12.34.20.575323 | ... |
| 0 | 0 | MAIN | EI | 2011-04-28-12.34.20.602452 | ... |
| 0 | 0 | MAIN | EI | 2011-04-28-12.34.20.665227 | ... |
| 0 | 0 | MAIN | EI | 2011-04-28-09.44.57.715392 | ... |

12 record(s) selected.

Output from this query (continued).

| ... | IMPACT | MSG |
|-------|--------|---|
| ----- | ----- | ----- |
| ... | - | - |
| ... | - | - |
| ... | - | Obtained exclusive mode lock on the file: |
| ... | - | - |
| ... | - | ADM7513W Database manager has started. |
| ... | - | Released lock on the file: |
| ... | - | - |
| ... | - | - |
| ... | - | Obtained exclusive mode lock on the file: |
| ... | - | ZRC=0xFFFFFBFE=-1026 |
| ... | - | Released lock on the file: |
| ... | - | - |

PDLOGMSGs_LAST24HOURS administrative view and PD_GET_LOG_MSGS table function - Retrieve problem determination messages

The PDLOGMSGs_LAST24HOURS administrative view and the PD_GET_LOG_MSGS table function return problem determination log messages that were logged in the database notification log.

The information returned is intended for use by database administrators and system administrators.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “PDLOGMSGSGS_LAST24HOURS administrative view”
- “PD_GET_LOG_MSGSGS table function” on page 1105

PDLOGMSGSGS_LAST24HOURS administrative view

The PDLOGMSGSGS_LAST24HOURS administrative view returns problem determination log messages that were logged in the database notification log in the last 24 hours.

The schema is SYSIBMADM.

Refer to Table 223 on page 1106 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- SELECT privilege on the PDLOGMSGSGS_LAST24HOURS administrative view
- CONTROL privilege on the PDLOGMSGSGS_LAST24HOURS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Get all critical log messages logged in the last 24 hours, ordered by most recent.

```
SELECT * FROM SYSIBMADM.PDLOGMSGSGS_LAST24HOURS
WHERE MSGSEVERITY = 'C' ORDER BY TIMESTAMP DESC
```

The following is an example of output from this query.

| TIMESTAMP | TIMEZONE | INSTANCENAME | MEMBER | ... |
|----------------------------|----------|--------------|--------|-----|
| 2005-11-23-21.56.41.240066 | -300 | svtdbm4 | 0 | ... |
| | | | | ... |
| | | | | ... |
| | | | | ... |
| | | | | ... |
| | | | | ... |
| | | | | ... |
| 2005-11-23-21.56.39.150597 | -300 | svtdbm4 | 0 | ... |
| 2005-11-23-21.56.37.363384 | -300 | svtdbm4 | 0 | ... |
| | | | | ... |
| | | | | ... |
| | | | | ... |
| 2005-11-23-21.56.35.880314 | -300 | svtdbm4 | 0 | ... |
| | | | | ... |

4 record(s) selected.

| DBPARTITIONNUM | DBNAME | PID | PROCESSNAME |
|----------------|---------|---------|----------------------|
| 0 | CAPTAIN | 4239374 | db2agent (CAPTAIN) 0 |
| 0 | CAPTAIN | 4239374 | db2agent (CAPTAIN) 0 |
| 0 | CAPTAIN | 4239374 | db2agent (CAPTAIN) 0 |

```

...TID APPL_ID                                COMPONENT                                ...
...-----
... 1 9.26.15.148.36942.051124025612 oper system services ...
...
...
...
...
...
...
...
... 1 9.26.15.148.36942.051124025612 base sys utilities ...
... 1 9.26.15.148.36942.051124025612 relation data serv ...
...
...
...
...
... 1 9.26.15.148.36942.051124025612 relation data serv ...
...
...
...

```

| ... | FUNCTION | PROBE | MSGNUM | MSGTYPE | ... |
|-----|-------------------|-------|--------|---------|-----|
| ... | ----- | ----- | ----- | ----- | ... |
| ... | sqloSleepInstance | 38 | 504 | ADM | ... |
| ... | | | | | ... |
| ... | | | | | ... |
| ... | | | | | ... |
| ... | | | | | ... |
| ... | | | | | ... |
| ... | | | | | ... |
| ... | sqlMarkDBad | 10 | 7518 | ADM | ... |
| ... | sqlrr_dump_ffdc | 10 | 1 | ADM | ... |
| ... | | | | | ... |
| ... | | | | | ... |
| ... | sqlrr_dump_ffdc | 10 | 1 | ADM | ... |
| ... | | | | | ... |

```
... MSGSEVERITY MSG
... -----
... C          ADM0504C An unexpected internal
...              processing error has occurred. ALL
...              Db2 PROCESSES ASSOCIATED WITH THIS
...              INSTANCE HAVE BEEN SUSPENDED.
...              Diagnostic information has been
```

```

...          recorded. Contact IBM Support
...          for further assistance.
... C        ADM7518C "CAPTAIN " marked bad.
... C        ADM0001C A severe error has occurred.
...          Examine the administration notification
...          log and contact IBM Support if
...          necessary.
... C        ADM0001C A severe error has occurred.
...          Examine the administration notification
...          log and contact IBM Support if necessary.

```

PD_GET_LOG_MSGS table function

The PD_GET_LOG_MSGS table function returns the same information as the PDLOGMSGs_LAST24HOURS administrative view, but allows you to specify a specific time period that is not limited to the last 24 hours.

Refer to Table 223 on page 1106 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the PD_GET_LOG_MSGS table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Syntax

```

▶▶ PD_GET_LOG_MSGS (—oldest_timestamp—, —member—) —————▶▶

```

The schema is SYSPROC.

Table function parameter

oldest_timestamp

An input argument of type **TIMESTAMP** that specifies a valid timestamp. Entries are returned starting with the most current timestamp and ending with the log entry with the timestamp specified by this input argument. If a null value is specified, all log entries are returned.

member

An optional input argument of type **INTEGER** that specifies a valid database member from which the records should be fetched from. Specify -1 or null for the current member, or -2 for information from all active database members. An active database member is where the database is available for connection and use by applications. If a cluster caching facility (CF) is specified, an active member will be used to request this data. If the notification log is not accessible, an error will be returned.

If the parameter is not specified, the default value is all active database members.

Information returned

Note: In a multi-member environment, the order in which log messages are returned cannot be guaranteed. If the order of log records is important, the results should be sorted by timestamp.

Table 223. Information returned by the PDLOGMSGs_LAST24HOURS administrative view and the PD_GET_LOG_MSGS table function

| Column name | Data type | Description |
|----------------|--------------|---|
| TIMESTAMP | TIMESTAMP | The time when the entry was logged. |
| TIMEZONE | INTEGER | Time difference (in minutes) from Universal Coordinated Time (UCT). For example, -300 is EST. |
| INSTANCENAME | VARCHAR(128) | Name of the instance that generated the message. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| DBNAME | VARCHAR(128) | The database on which the error or event occurred. |
| PID | BIGINT | Process ID of the process that generated the message. |
| PROCESSNAME | VARCHAR(255) | Name of process that generated the message. |
| TID | BIGINT | ID of the thread within the process that generated the message. |
| APPL_ID | VARCHAR(64) | appl_id - Application ID monitor element |
| COMPONENT | VARCHAR(255) | The name of the database component that is providing the message. For messages written by user applications using the db2AdminMsgWrite API, "User Application" is returned. |
| FUNCTION | VARCHAR(255) | The name of the database function that is providing the message. For messages written by user applications using the db2AdminMsgWrite API, "User Function" is returned. |

Table 223. Information returned by the PDLOGMSGG_LAST24HOURS administrative view and the PD_GET_LOG_MSGG table function (continued)

| Column name | Data type | Description |
|-------------|-----------|---|
| PROBE | INTEGER | Unique internal identifier that allows Customer Support and Development to locate the point in the source code that generated the message. |
| MSGNUM | INTEGER | The numeric message number for the error or event. |
| MSGTYPE | CHAR(3) | Indicates the message type: ADM (for messages written to the administration notification log) or NULL if the message type cannot be determined. |
| MSGSEVERITY | CHAR(1) | Message severity: C (critical), E (error), W (warning), I (informational) or NULL (if the message severity could not be determined). |
| MSG | CLOB(16K) | Notification log message text. |
| MEMBER | SMALLINT | member - Database member monitor element |

Examples

Example 1: Retrieve all notification messages logged for database SAMPLE on instance XYZ in the last week for all database partitions. Report messages in chronological order.

```
SELECT TIMESTAMP, APPL_ID, DBPARTITIONNUM, MSG
FROM TABLE ( PD_GET_LOG_MSGG( CURRENT_TIMESTAMP - 7 DAYS)) AS T
WHERE INSTANCENAME = 'XYZ' AND DBNAME = 'SAMPLE'
ORDER BY TIMESTAMP ASC
```

The following is an example of output from this query.

| TIMESTAMP | APPL_ID | DBPARTITIONNUM ... |
|----------------------------|-------------------------|--------------------|
| 2005-11-13-12.51.37.772000 | *LOCAL.DB2.050324175005 | 0 ... |
| | | ... |
| 2005-11-13-12.51.37.772001 | *LOCAL.DB2.050324175005 | 0 ... |
| | | ... |
| 2005-11-13-12.51.37.781000 | *LOCAL.DB2.050324175005 | 0 ... |
| | | ... |
| 2005-11-13-12.51.37.781001 | *LOCAL.DB2.050324175005 | 0 ... |
| | | ... |
| | | ... |
| 2005-11-17-14.12.39.036001 | *LOCAL.DB2.041117191249 | 0 ... |
| 2005-11-17-14.12.39.056000 | *LOCAL.DB2.041117191249 | 0 ... |
| 2005-11-17-14.13.04.450000 | *LOCAL.DB2.041117191307 | 0 ... |
| 2005-11-17-14.13.04.460000 | *LOCAL.DB2.041117191307 | 0 ... |
| 2005-11-17-14.18.29.042000 | *LOCAL.DB2.041117190824 | 0 ... |
| ... | | |
| ... | | |
| ... | | |

Output from this query (continued).

```
... MSG
... -----...--
... ADM5502W The escalation of "143" locks on table
... "SYSIBM .SYSINDEXAUTH" to lock intent "X" was successful.
... ADM5502W The escalation of "144" locks on table
... "SYSIBM .SYSINDEXES" to lock intent "X" was successful.
... ADM5502W The escalation of "416" locks on table
... "SYSIBM .SYSINDEXCOLUSE" to lock intent "X" was successful.
... ADM5500W Db2 is performing lock escalation. The total
... number of locks currently held is "1129", and the target
... number of locks to hold is "564".
... ADM7506W Database quiesce has been requested.
... ADM7507W Database quiesce request has completed successfully.
... ADM7510W Database unquiesce has been requested.
... ADM7509W Database unquiesce request has completed successfully.
... ADM4500W A package cache overflow condition has occurred. There
... is no error but this indicates that the package cache has
... exceeded the configured maximum size. If this condition persists,
... you may want to adjust the PCKCACHESZ DB configuration parameter.
```

Example 2: Retrieve all critical errors logged on instance XYZ for database partition 0 in the last day, sorted by most recent.

```
SELECT TIMESTAMP, DBNAME, MSG
  FROM TABLE (PD_GET_LOG_MSGS(CURRENT_TIMESTAMP - 1 DAYS)) AS T
 WHERE MSGSEVERITY = 'C' AND INSTANCENAME = 'XYZ' AND
    DBPARTITIONNUM = 0
 ORDER BY TIMESTAMP DESC
```

The following is an example of output from this query.

| TIMESTAMP | DBNAME | MSG |
|----------------------------|----------|--|
| 2004-11-04-13.49.17.022000 | TESTSBCS | ADM0503C An unexpected internal processing error has occurred. ALL Db2 PROCESSES ASSOCIATED WITH THIS INSTANCE HAVE BEEN SHUTDOWN. Diagnostic information has been recorded. Contact IBM Support for further assistance. |
| 2004-11-04-11.32.26.760000 | SAMPLE | ADM0503C An unexpected internal processing error has occurred. ALL Db2 PROCESSES ASSOCIATED WITH THIS INSTANCE HAVE BEEN SHUTDOWN. Diagnostic information has been recorded. Contact IBM Support for further assistance. |

2 record(s) selected.

Example 3: Retrieve messages written by database processes servicing application with application ID of *LOCAL.DB2.050927195337, over the last day.

```
SELECT TIMESTAMP, MSG
  FROM TABLE (PD_GET_LOG_MSGS(CURRENT_TIMESTAMP - 1 DAYS)) AS T
 WHERE APPL_ID = '*LOCAL.DB2.050927195337'
```

The following is an example of output from this query.

| TIMESTAMP | MSG |
|----------------------------|---|
| 2005-06-27-21.17.12.389000 | ADM4500W A package cache overflow condition has occurred. There is no error but this indicates that the package cache has exceeded the configured maximum size. If this condition persists, you may want to adjust the PCKCACHESZ DB configuration parameter. |
| 2005-06-27-18.41.22.248000 | ADM4500W A package cache overflow condition has occurred. There is no error but this indicates that the package cache has exceeded the configured maximum size. If this condition persists, you may want to adjust the PCKCACHESZ DB configuration parameter. |
| 2005-06-27-12.51.37.772001 | ADM5502W The escalation of "143" locks on table "SYSIBM .SYSINDEXAUTH" to lock intent "X" was successful. |
| 2005-06-27-12.51.37.772000 | ADM5502W The escalation of "144" locks on table "SYSIBM .SYSINDEXES" to lock intent "X" was successful. |
| 2005-06-27-12.51.37.761001 | ADM5502W The escalation of "416" locks on table "SYSIBM .SYSINDEXCOLUSE" to lock intent "X" was successful. |
| ... | |

Example 4: Find all instances of message ADM0504C in the notification log. Note that the messages considered are not limited by a timestamp. This could be an expensive operation if the notification logfile is very large.

```
SELECT TIMESTAMP, DBPARTITIONNUM, DBNAME, MSG
FROM TABLE (PD_GET_LOG_MSGS(CAST(NULL AS TIMESTAMP))) AS T
WHERE MSGNUM = 504 AND MSGTYPE = 'ADM' AND MSGSEVERITY = 'C'
```

The following is an example of output from this query.

| TIMESTAMP | DBPARTITIONNUM | DBNAME | ... |
|----------------------------|----------------|---------|-----|
| 2005-11-23-21.56.41.240066 | 0 | CAPTAIN | ... |
| ... | | | |
| ... | | | |
| ... | | | |
| ... | | | |
| ... | | | |
| ... | | | |
| ... | | | |
| ... | | | |

Output from this query (continued).

| ... | APPL_ID | MSG |
|-----|--------------------------------|---|
| ... | 9.26.15.148.36942.051124025612 | ADM0504C An unexpected internal processing error has occurred. ALL Db2 PROCESSES ASSOCIATED WITH THIS INSTANCE HAVE BEEN SUSPENDED. Diagnostic information has been recorded. Contact IBM Support for further assistance. |
| ... | | |
| ... | | |
| ... | | |
| ... | | |
| ... | | |
| ... | | |
| ... | | |
| ... | | |

Example 5: Retrieve all notification messages for a specific member.

```
SELECT MEMBER,DBPARTITIONNUM, TIMESTAMP, SUBSTR(MSG,1, 50)
FROM TABLE (PD_GET_LOG_MSGS(NULL,-2))
```

The following is an example of output from this query.

| MEMBER | DBPARTITIONNUM | TIMESTAMP | ... |
|--------|----------------|----------------------------|-----|
| 0 | 0 | 2011-04-27-09.51.17.725916 | ... |
| 2 | 2 | 2011-04-27-09.51.16.801966 | ... |
| 1 | 1 | 2011-04-27-09.51.16.747745 | ... |
| 3 | 3 | 2011-04-27-09.51.15.655162 | ... |

4 record(s) selected.

Output from this query (continued).

```
... 4
... -----
... ADM7513W Database manager has started.
...
... ADM7513W Database manager has started.
...
... ADM7513W Database manager has started.
...
... ADM7513W Database manager has started.
```

REORGCHK_IX_STATS procedure - Retrieve index statistics for reorganization evaluation

The REORGCHK_IX_STATS procedure returns a result set containing index statistics that indicate whether or not there is a need for reorganization.

Syntax

```
►►—REORGCHK_IX_STATS—(—scope—,—criteria—)—————►►
```

The schema is SYSPROC.

Procedure parameters

scope

An input argument of type CHAR(1) that specifies the scope of the tables that are to be evaluated, using one of the following values:

'T'

Table

'S'

Schema

criteria

An input argument of type VARCHAR(259).

If *scope* has a value of 'T', use this argument to specify one of the following values:

- A fully qualified table name
- The value ALL for all tables
- The value USER for all user-defined tables

- The value SYSTEM for system-defined tables

If *scope* has a value of 'S', use this argument to specify a schema name.

Authorization

- SELECT privilege on catalog tables.
- EXECUTE privilege on the REORGCHK_IX_STATS procedure.

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

```
CALL SYSPROC.REORGCHK_IX_STATS('T','JESCOTT.EMPLOYEE')
```

Usage note

The procedure uses the SYSTOOLSTMPSPACE table space. If SYSTOOLSTMPSPACE does not already exist, the procedure will create this table space.

Information returned

Table 224. Information returned by the REORGCHK_IX_STATS procedure

| Column name | Data type | Description |
|-------------------|--------------|---|
| TABLE_SCHEMA | VARCHAR(128) | table_schema - Table schema name monitor element |
| TABLE_NAME | VARCHAR(128) | table_name - Table name monitor element |
| INDEX_SCHEMA | VARCHAR(128) | index_schema - Index schema monitor element |
| INDEX_NAME | VARCHAR(128) | index_name - Index name monitor element |
| DATAPARTITIONNAME | VARCHAR(128) | Name of the data partition. NULL for nonpartitioned tables. |
| INDCARD | BIGINT | Number of index entries in the index. This can be different than table cardinality for some indexes. For example, the index cardinality on XML columns might be greater than the table cardinality. |
| NLEAF | BIGINT | nleaf - Number of leaf pages monitor element |
| NUM_EMPTY_LEAFS | BIGINT | Number of pseudo-empty index leaf pages. |
| NLEVELS | INTEGER | nlevels - Number of index levels monitor element |
| NUMRIDS_DELETED | BIGINT | Number of pseudo-deleted RIDs. |
| FULLKEYCARD | BIGINT | Number of unique index entries that are not marked deleted. |

Table 224. Information returned by the REORGCHK_IX_STATS procedure (continued)

| Column name | Data type | Description |
|-----------------------|-----------|---|
| LEAF_REC_SIZE | BIGINT | Record size of the index entry on a leaf page. This is the average size of the index entry excluding any overhead and is calculated from the average column length of all columns participating in the index. |
| NONLEAF_REC_SIZE | BIGINT | Record size of the index entry on a non-leaf page. This is the average size of the index entry excluding any overhead and is calculated from the average column length of all columns participating in the index except any INCLUDE columns. |
| LEAF_PAGE_OVERHEAD | BIGINT | Reserved space on the index leaf page for internal use. |
| NONLEAF_PAGE_OVERHEAD | BIGINT | Reserved space on the index non-leaf page for internal use |
| PCT_PAGES_SAVED | SMALLINT | Percent of pages saved using Index Compression. A non-zero number indicates the index is compressed. |
| F4 | INTEGER | F4 formula value. |
| F5 | INTEGER | F5 formula value. |
| F6 | INTEGER | F6 formula value. |
| F7 | INTEGER | F7 formula value. |
| F8 | INTEGER | F8 formula value. |
| REORG | CHAR(5) | A 5-character field, each character mapping to one of the five formulas: F4, F5, F6, F7, and F8; a dash means that the formula value is in the recommended range; an asterisk means that the formula value is out of the recommended range, indicating a need for reorganization. |

REORGCHK_TB_STATS procedure - Retrieve table statistics for reorganization evaluation

The REORGCHK_TB_STATS procedure returns a result set containing table statistics that indicate whether or not there is a need for reorganization.

Syntax

►► REORGCHK_TB_STATS (—scope—, —criteria—) ◀◀

The schema is SYSPROC.

Procedure parameters

scope

An input argument of type CHAR(1) that specifies the scope of the tables that are to be evaluated, using one of the following values:

'T'

Table

'S'

Schema

criteria

An input argument of type VARCHAR(259).

If *scope* has a value of 'T', use this argument to specify one of the following values:

- A fully qualified table name
- The value ALL for all tables
- The value USER for all user-defined tables
- The value SYSTEM for system-defined tables

If *scope* has a value of 'S', use this argument to specify a schema name.

Authorization

- SELECT privilege on catalog tables.
- EXECUTE privilege on the REORGCHK_TB_STATS procedure.

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

```
CALL SYSPROC.REORGCHK_TB_STATS('T','JESCOTT.EMPLOYEE')
```

Usage note

The procedure uses the SYSTOOLSTMPSPACE table space. If SYSTOOLSTMPSPACE does not already exist, the procedure will create this table space.

Information returned

Table 225. Information returned by the REORGCHK_TB_STATS procedure

| Column name | Data type | Description |
|-------------------|--------------|---|
| TABLE_SCHEMA | VARCHAR(128) | table_schema - Table schema name monitor element |
| TABLE_NAME | VARCHAR(128) | table_name - Table name monitor element |
| DATAPARTITIONNAME | VARCHAR(128) | Name of the data partition. NULL for nonpartitioned tables. |
| CARD | BIGINT | Cardinality (number of rows in the table). |
| OVERFLOW | BIGINT | Number of overflow rows. |

Table 225. Information returned by the REORGCHK_TB_STATS procedure (continued)

| Column name | Data type | Description |
|---------------|-----------|---|
| NPAGES | BIGINT | Total number of pages on which the rows of the table exist; -1 for a view or alias, or if statistics are not collected; -2 for a subtable or hierarchy table. |
| FPAGES | BIGINT | Total number of pages; -1 for a view or alias, or if statistics are not collected; -2 for a subtable or hierarchy table. |
| ACTIVE_BLOCKS | BIGINT | Total number of active blocks for a multidimensional clustering (MDC) or insert time clustering (ITC) table. This field is only applicable to tables defined using the ORGANIZE BY clause. It indicates the number of blocks of the table that contains data. |
| TSIZE | BIGINT | Size of the table. |
| F1 | INTEGER | F1 formula value. |
| F2 | INTEGER | F2 formula value. |
| F3 | INTEGER | F3 formula value. |
| REORG | CHAR(3) | A 3-character field, each character mapping to one of the three formulas: F1, F2, and F3; a dash means that the formula value is in the recommended range; an asterisk means that the formula value is out of the recommended range, indicating a need for reorganization |

SET_MAINT_MODE_RECORD_NO_TEMPORALHISTORY procedure - disable recording of temporal history

The SET_MAINT_MODE_RECORD_NO_TEMPORALHISTORY procedure sets the internal option to disable recording of temporal history for a system-period temporal table and allows values to be specified for the special temporal columns.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the routine

SECADM authority is required to grant EXECUTE privilege on the SET_MAINT_MODE_RECORD_NO_TEMPORALHISTORY procedure. EXECUTE privilege WITH GRANT OPTION cannot be granted for this routine (SQLSTATE 42501).

Default PUBLIC privilege

None

Syntax

►►—SET_MAINT_MODE_RECORD_NO_TEMPORALHISTORY—(—)—————►►

The schema is SYSPROC.

Usage notes

- Disabling the recording of temporal history for a system-period temporal table with the SET_MAINT_MODE_RECORD_NO_TEMPORALHISTORY procedure lasts for the duration of the current session.

Example

Set the internal option to disable recording of temporal history for a system-period temporal table and allow values to be specified for the special temporal columns.

```
call SYSPROC.SET_MAINT_MODE_RECORD_NO_TEMPORALHISTORY()
```

SQLERRM scalar functions - Retrieves error message information

There are two versions of the SQLERRM scalar function. The first allows for full flexibility of message retrieval including using message tokens and language selection. The second takes only an SQLCODE as an input parameter and returns the short message in English.

SQLERRM scalar function

This SQLERRM scalar function takes a message identifier, locale and token input and returns the short or long message in the specified locale. If the input locale is not supported by the server, the message is returned in English.

Syntax

►►—SQLERRM—(—msgid—,—tokens—,—token_delimiter—,—locale—,—shortmsg—►►
►—)—————►►

The schema is SYSPROC.

Scalar function parameters

msgid

An input argument of type VARCHAR(9) that represents the message number for which the information should be retrieved. The message number is the application return code prefixed with 'SQL', 'DBA' or 'CLI'. For example, 'SQL551', 'CLI0001'. The message number can also be an SQLSTATE, for example, '42829'.

tokens

An input argument of type VARCHAR(70) that represents the error message token list. Some messages might not have tokens. If this parameter is null, then no token replacement occurs in the returned message. Token replacement only occurs when returning the default short messages. If the long message option is selected, no token replacement occurs.

token_delimiter

An input argument of type VARCHAR(1) that represents the token delimiter. This delimiter must be unique and not contained in any tokens passed to the scalar function. If no delimiter is supplied, the default delimiter used is the semicolon.

locale

An input argument of type VARCHAR(33) that represents the locale to pass to the server in order to have the error message retrieved in that language. If no locale is specified, or the server does not support the locale, the message is returned in English and a warning is returned.

shortmsg

An input argument of type INTEGER that is used to indicate if the long message should be returned instead of the default short message. To return long messages, this value must be set to 0 or CAST(NULL as INTEGER).

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Information returned

The data type of the result is VARCHAR(32672).

Examples

Example 1: Retrieve the English short message for SQL0551N with tokens "AYYANG", "UPDATE" and "SYSCAT.TABLES".

```
VALUES (SYSPROC.SQLERRM
        ('SQL551', 'AYYANG;UPDATE;SYSCAT.TABLES', ';', 'en_US', 1))
```

The following is an example of output returned.

```
1
-----
SQL0551N "AYYANG" does not have the privilege to perform operation
      "UPDATE" on object "SYSCAT.TABLES"
```

Example 2: Retrieve the English error message associated with SQLSTATE 42501.

```
VALUES (SYSPROC.SQLERRM ('42501', '', '', 'en_US', 1))
```

The following is an example of output returned.

```
1
-----
SQLSTATE 42501: The authorization ID does not have the privilege to
perform the specified operation on the identified object.
```

Example 3: Retrieve the English long error message for SQL1001N.
VALUES (SYSPROC.SQLERRM ('SQL1001', ' ', ' ', 'en_US', 0))

The following is an example of output returned.

```
1
-----
SQL1001N "<name>" is not a valid database name.
```

Explanation:

The syntax of the database name specified in the command is not valid. The database name must contain 1 to 8 characters and all the characters must be from the database manager base character set.

The command cannot be processed.

User Response:

Resubmit the command with the correct database name.

sqlcode : -1001

sqlstate : 2E000

SQLERRM scalar function

This SQLERRM scalar function takes an SQLCODE as the only input and returns the short message for the specified SQLCODE in English.

Syntax

►►—SQLERRM—(—*sqlcode*—)—————►►

The schema is SYSPROC.

Scalar function parameter

sqlcode

An input argument of type INTEGER that represents an SQLCODE.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Information returned

The data type of the result is VARCHAR(32672).

Example

Retrieve the short message for SQLCODE SQL0551N.

```
VALUES (SYSPROC.SQLERRM (551))
```

The following is an example of output returned.

```
1
-----...--
SQL0551N  "" does not have the privilege to perform operation
         "" on object "".
```

SYSINSTALLOBJECTS

The SYSINSTALLOBJECTS procedure creates or drops the database objects that are required for a specific tool.

Syntax

```
►—SYSINSTALLOBJECTS—(—tool-name—,—action—,—tablespace-name—,——————►
►—schema-name—)—————►◄
```

The schema is SYSPROC.

Procedure parameters

tool-name

An input argument of type VARCHAR(128) that specifies the name of the tool that is to be loaded, using one of the following values:

- 'DB2AC' for autonomous computing (health monitor)
- 'STMG_DBSIZE_INFO' for storage management
- 'OPT_PROFILES' for creating the optimization profile table
- 'POLICY' for policy (tables and triggers)
- 'EXPLAIN' for creating or migrating explain tables
- 'INGEST' for creating the restart table used by the ingest utility
- 'REPL_MQT' for creating the latency table for shadow tables

action

An input argument of type CHAR(1) that specifies the action that is to be taken. Valid values are:

- | | |
|----------|--|
| C | Create objects. |
| D | Drop objects. |
| V | Verify objects. |
| M | Migrate objects. The M option is only valid when used with the tool name EXPLAIN. This option migrates explain tables that were created in Version 7 or later to be compatible with the current version. |

tablespace-name

An input argument of type VARCHAR(128) that specifies the name of the table space in which the objects are to be created. If a value is not specified, or the value is an empty or blank string, the default user space is used if the tool name is AM. If the tool name is EXPLAIN and the action is M, the input table space name is ignored and the table space is used where the explain tables that

are being migrated were created. Otherwise, the SYSTOOLSPACE table space is used. If SYSTOOLSPACE does not already exist, it will be created.

schema-name

Except for 'EXPLAIN' tool-name option, SYSTOOLS is always used as the schema regardless of the schema-name passed as the input parameter.

For 'EXPLAIN' tool-name option, an input schema-name can be passed and the tables are created under the specified schema-name. If no schema-name is passed as the input parameter, SYSTOOLS schema is used.

Authorization

One of the following authorities is required to execute the procedure:

- EXECUTE privilege on the procedure
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Examples

1. Migrate all explain tables.

```
CALL SYSPROC.SYSINSTALLOBJECTS('EXPLAIN', 'M', CAST (NULL AS VARCHAR(128)),  
    CAST (NULL AS VARCHAR(128)))
```

2. Verify objects that are used for autonomous computing (health monitor).

```
CALL SYSPROC.SYSINSTALLOBJECTS('DB2AC', 'V', CAST (NULL AS VARCHAR(128)),  
    CAST (NULL AS VARCHAR(128)))
```

This CALL statement results in the following message:

```
SQL0438N  
Application raised error or warning with diagnostic text: "DB2AC INVALID".  
SQLSTATE=3900
```

Deprecated SQL administrative routines and views

Expanded support is provided for the existing administrative routines and views by replacing them with new, more comprehensive routines or views.

Starting with Version 10.1, routines with version-suffixed names are deprecated. The names of the routines do not have a version suffix so that the names remain consistent across releases. The replacement routines without a version-suffix might include modifications such as: columns added or removed, new data types for existing columns, or new values for existing columns. Use good practices when issuing queries on SQL administrative routines to minimize the impact from changes to these routines.

The following table provides a list of all the SQL administrative routines or views that are currently deprecated:

Table 226. Deprecated SQL administrative routines or views and their replacement routines or views

| Deprecated routine or view | Deprecated since | New routine or view | Replacement available since |
|--|------------------|--|-----------------------------|
| GET_ROUTINE_SAR procedure | Version 11.1 | Not available | |
| "APPL_PERFORMANCE administrative view - Retrieve percentage of rows selected for an application" on page 1165 | Version 10.5 | "MON_CONNECTION_SUMMARY - Retrieve metrics for all connections" on page 430 | |
| "APPLICATIONS administrative view - Retrieve connected database application information" on page 1167 | Version 10.5 | "MON_GET_CONNECTION table function - Get connection metrics" on page 588 | |
| "BP_HITRATIO administrative view - Retrieve bufferpool hit ratio information" on page 1172 | Version 10.5 | "MON_BP_UTILIZATION administrative view - Retrieve metrics for bufferpools" on page 422 | |
| "BP_READ_IO administrative view - Retrieve bufferpool read performance information" on page 1174 | Version 10.5 | "MON_BP_UTILIZATION administrative view - Retrieve metrics for bufferpools" on page 422 | |
| "BP_WRITE_IO administrative view - Retrieve bufferpool write performance information" on page 1176 | Version 10.5 | "MON_BP_UTILIZATION administrative view - Retrieve metrics for bufferpools" on page 422 | |
| "CONTAINER_UTILIZATION administrative view - Retrieve table space container and utilization information" on page 1178 | Version 10.5 | "MON_GET_CONTAINER table function - Get table space container metrics" on page 616 | |
| "LOG_UTILIZATION administrative view - Retrieve log utilization information" on page 1242 | Version 10.5 | "MON_TRANSACTION_LOG_UTILIZATION administrative view - Retrieve log utilization information" on page 454 | Version 10.5 |
| "LONG_RUNNING_SQL administrative view" on page 1244 | Version 10.5 | "MON_CURRENT_SQL - Retrieve key metrics for all activities on all members" on page 435 | |
| "SNAP_WRITE_FILE procedure" on page 1428 | Version 10.5 | Not available | |
| "SNAPAGENT administrative view and SNAP_GET_AGENT table function - Retrieve agent logical data group application snapshot information" on page 1250 | Version 10.5 | "MON_GET_CONNECTION table function - Get connection metrics" on page 588 and "MON_GET_AGENT table function - List agents, fenced mode processes, and system entities for the database" on page 551 | |
| "SNAPAPPL administrative view and SNAP_GET_APPL table function - Retrieve appl logical data group snapshot information" on page 1272 | Version 10.5 | "MON_GET_CONNECTION table function - Get connection metrics" on page 588 | |
| "SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function - Retrieve appl_info logical data group snapshot information" on page 1258 | Version 10.5 | "MON_GET_CONNECTION table function - Get connection metrics" on page 588 | |
| "SNAPBP administrative view and SNAP_GET_BP table function - Retrieve bufferpool logical group snapshot information" on page 1287 | Version 10.5 | "MON_GET_BUFFERPOOL table function - Get buffer pool metrics" on page 571 | |
| "SNAPBP_PART administrative view and SNAP_GET_BP_PART table function - Retrieve bufferpool_nodeinfo logical data group snapshot information" on page 1293 | Version 10.5 | "MON_GET_BUFFERPOOL table function - Get buffer pool metrics" on page 571 | |
| "SNAPCONTAINER administrative view and SNAP_GET_CONTAINER table function - Retrieve tablespace_container logical data group snapshot information" on page 1300 | Version 10.5 | "MON_GET_CONTAINER table function - Get table space container metrics" on page 616 | |
| "SNAPDB administrative view and SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group" on page 1308 | Version 10.5 | "MON_GET_DATABASE table function - Get database metrics" on page 619 | Version 10.5 |
| "SNAPDBM administrative view and SNAP_GET_DBM table function - Retrieve the dbm logical grouping snapshot information" on page 1334 | Version 10.5 | "MON_GET_INSTANCE table function - Get instance level information" on page 667 | Version 10.5 |
| "SNAPFCM administrative view and SNAP_GET_FCM table function - Retrieve the fcm logical data group snapshot information" on page 1355 | Version 10.5 | "MON_GET_FCM - Get FCM metrics" on page 652 | |
| "SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG table function - Retrieve snapshot information from the detail_log logical data group" on page 1345 | Version 10.5 | "MON_GET_TRANSACTION_LOG table function - Get log information" on page 827 | Version 10.5 |

Table 226. Deprecated SQL administrative routines or views and their replacement routines or views (continued)

| Deprecated routine or view | Deprecated since | New routine or view | Replacement available since |
|---|------------------|---|-----------------------------|
| "SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function - Retrieve the fcm_node logical data group snapshot information" on page 1358 | Version 10.5 | "MON_GET_FCM - Get FCM metrics" on page 652 | |
| "SNAPSTMT administrative view and SNAP_GET_STMT table function - Retrieve statement snapshot information" on page 1382 | Version 10.5 | "MON_CURRENT_SQL - Retrieve key metrics for all activities on all members" on page 435 and "MON_GET_CONNECTION table function - Get connection metrics" on page 588 | Version 9.7 |
| "SNAPTbsp administrative view and SNAP_GET_TBSP table function - Retrieve table space logical data group snapshot information" on page 1399 | Version 10.5 | "MON_GET_TABLESPACE table function - Get table space metrics" on page 805 | |
| "SNAPTbsp_PART administrative view and SNAP_GET_TBSP_PART table function - Retrieve tablespace_nodeinfo logical data group snapshot information" on page 1405 | Version 10.5 | "MON_GET_TABLESPACE table function - Get table space metrics" on page 805 | |
| "SNAPTbsp_QUIESCER administrative view and SNAP_GET_TBSP_QUIESCER table function - Retrieve quiescer table space snapshot information" on page 1419 | Version 10.5 | "MON_GET_TABLESPACE_QUIESCER table function - Get information about quiesced table spaces" on page 817 | Version 10.5 Fix Pack 3 |
| "SNAPTbsp_RANGE administrative view and SNAP_GET_TBSP_RANGE table function - Retrieve range snapshot information" on page 1377 | Version 10.5 | "MON_GET_TABLESPACE_RANGE table function - Get information about table space ranges" on page 820 | Version 10.5 Fix Pack 3 |
| "SNAPUTIL administrative view and SNAP_GET_UTIL table function - Retrieve utility_info logical data group snapshot information" on page 1424 | Version 10.5 | "MON_GET_UTILITY table function - Get utilities running on the database" on page 863 | Version 10.5 |
| "TBSP_UTILIZATION administrative view - Retrieve table space configuration and utilization information" on page 1572 | Version 10.5 | "MON_TBSP_UTILIZATION - Retrieve monitoring metrics for all table spaces and all database partitions" on page 449 | |
| "ADMIN_GET_DBP_MEM_USAGE table function - Get total memory consumption for instance" on page 1124 | Version 10.1 | "ADMIN_GET_MEM_USAGE table function - Get total memory consumption for instance" on page 214 | Version 10.1 |
| "ADMIN_TABCOMPRESSINFO administrative view and ADMIN_GET_TAB_COMPRESS_INFO table function (deprecated) - returns compressed information" on page 1126 | Version 10.1 | "ADMIN_GET_TAB_COMPRESS_INFO table function - estimate compression savings" on page 219 and "ADMIN_GET_TAB_DICTIONARY_INFO table function - report properties of existing table dictionaries" on page 221 | Version 10.1 |
| "ADMIN_GET_TAB_COMPRESS_INFO_V97" on page 1132 | Version 10.1 | "ADMIN_GET_TAB_COMPRESS_INFO table function - estimate compression savings" on page 219 and "ADMIN_GET_TAB_DICTIONARY_INFO table function - report properties of existing table dictionaries" on page 221 | Version 10.1 |
| "ADMIN_GET_TAB_INFO_V95 table function - Retrieve size and state information for tables" on page 1138 | Version 10.1 | "ADMIN_GET_TAB_INFO table function" on page 225 | Version 10.1 |
| "ADMIN_GET_TAB_INFO_V97 table function - Retrieve size and state information for tables" on page 1143 | Version 10.1 | "ADMIN_GET_TAB_INFO table function" on page 225 | Version 10.1 |
| "AM_BASE_RPT_RECOMS - Recommendations for activity reports" on page 1150 | Version 10.1 | Not available | Version 10.1 |
| "AM_BASE_RPTS - Activity event monitor reports" on page 1152 | Version 10.1 | Not available | Version 10.1 |
| "AM_DROP_TASK - Delete a monitoring task" on page 1153 | Version 10.1 | Not available | Version 10.1 |
| "AM_GET_LOCK_CHN_TB - Retrieve application lock chain data in a tabular format" on page 1154 | Version 10.1 | Not available | Version 10.1 |
| "AM_GET_LOCK_CHNS - Retrieve lock chain information for a specific application" on page 1155 | Version 10.1 | Not available | Version 10.1 |
| "AM_GET_LOCK_RPT - Retrieve application lock details" on page 1156 | Version 10.1 | Not available | Version 10.1 |

Table 226. Deprecated SQL administrative routines or views and their replacement routines or views (continued)

| Deprecated routine or view | Deprecated since | New routine or view | Replacement available since |
|---|------------------|---|-----------------------------|
| "AM_GET_RPT - Retrieve activity monitor data" on page 1163 | Version 10.1 | Not available | Version 10.1 |
| "AM_SAVE_TASK - Create or modify a monitoring task" on page 1164 | Version 10.1 | Not available | Version 10.1 |
| "APPLICATION_ID" on page 1171 | Version 10.1 | "MON_GET_APPLICATION_ID - Get connection application ID" on page 565 | Version 10.1 |
| "REG_VARIABLES administrative view - Retrieve database registry settings in use" on page 1246 | Version 10.1 | "ENV_GET_REG_VARIABLES table function - Retrieve database registry settings that are in use" on page 389 | Version 10.1 |
| "SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function - Retrieve memory_pool logical data group snapshot information" on page 1254 | Version 10.1 | "MON_GET_MEMORY_POOL - get memory pool information" on page 679 and "MON_GET_MEMORY_SET - get memory set information" on page 682 | Version 10.1 |
| "SNAP_GET_APPL_INFO_V95 table function - Retrieve appl_info logical data group snapshot information" on page 1266 | Version 10.1 | "MON_GET_CONNECTION table function - Get connection metrics" on page 588 | Version 10.1 |
| "SNAP_GET_APPL_V95 table function - Retrieve appl logical data group snapshot information" on page 1280 | Version 10.1 | "MON_GET_CONNECTION table function - Get connection metrics" on page 588 | Version 10.1 |
| "SNAP_GET_BP_V95 table function - Retrieve bufferpool logical group snapshot information" on page 1297 | Version 10.1 | "MON_GET_BUFFERPOOL table function - Get buffer pool metrics" on page 571 | Version 10.1 |
| "SNAP_GET_CONTAINER_V91 table function - Retrieve tablespace_container logical data group snapshot information" on page 1305 | Version 10.1 | "MON_GET_CONTAINER table function - Get table space container metrics" on page 616 | Version 10.1 |
| "SNAP_GET_DB_V97 table function - Retrieve snapshot information from the dbase logical group" on page 1324 | Version 10.1 | "MON_GET_DATABASE table function - Get database metrics" on page 619 | Version 10.1 |
| "SNAP_GET_DBM_V95 table function - Retrieve the dbm logical grouping snapshot information" on page 1342 | Version 10.1 | "MON_GET_INSTANCE table function - Get instance level information" on page 667 | Version 10.1 |
| "SNAP_GET_DETAILLOG_V91 table function - Retrieve snapshot information from the detail_log logical data group" on page 1349 | Version 10.1 | MON_GET_TRANSACTION_LOG table function | Version 10.1 |
| "SNAP_GET_DYN_SQL_V95 table function - Retrieve dynsql logical group snapshot information" on page 1351 | Version 10.1 | "SNAP_GET_DYN_SQL table function" on page 942 | Version 10.1 |
| "SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS_V97 table function - Retrieve automatic storage path information" on page 1390 | Version 10.1 | "ADMIN_GET_STORAGE_PATHS table function - retrieve automatic storage path information" on page 217 | Version 10.1 |
| "SNAPTAB administrative view and SNAP_GET_TAB table function - Retrieve table logical data group snapshot information" on page 1393 | Version 10.1 | "MON_GET_BUFFERPOOL table function - Get buffer pool metrics" on page 571, "MON_GET_TABLE table function - get table metrics" on page 800, "MON_GET_TABLESPACE table function - Get table space metrics" on page 805, and "MON_BP_UTILIZATION administrative view - Retrieve metrics for bufferpools" on page 422 | Version 10.1 |
| "SNAP_GET_TAB_V91" on page 1397 | Version 10.1 | "MON_GET_TABLE table function - get table metrics" on page 800 | Version 10.1 |
| "SNAP_GET_TBSP_PART_V97 table function - Retrieve tablespace_nodeinfo logical data group snapshot information" on page 1411 | Version 10.1 | "MON_GET_TABLESPACE table function - Get table space metrics" on page 805 | Version 10.1 |
| "SNAP_GET_TBSP_V91" on page 1415 | Version 10.1 | "MON_GET_TABLESPACE table function - Get table space metrics" on page 805 | Version 10.1 |
| "SNAPHADR administrative view and SNAP_GET_HADR table function - Retrieve hadr logical data group snapshot information" on page 1361 | Version 10.1 | "MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information" on page 656 | Version 10.1 |
| "WLM_GET_SERVICE_CLASS_AGENTS_V97 - List agents running in a service class" on page 1580 | Version 10.1 | "MON_GET_AGENT table function - List agents, fenced mode processes, and system entities for the database" on page 551 | Version 10.1 |

Table 226. Deprecated SQL administrative routines or views and their replacement routines or views (continued)

| Deprecated routine or view | Deprecated since | New routine or view | Replacement available since |
|---|------------------|--|-----------------------------|
| "WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97 - List of workload occurrences" on page 1588 | Version 10.1 | "WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES - list workload occurrences" on page 1064 | Version 10.1 |
| "WLM_GET_SERVICE_SUBCLASS_STATS_V97 - return statistics of service subclasses" on page 1591 | Version 10.1 | "WLM_GET_SERVICE_SUBCLASS_STATS table function - Return statistics of service subclasses" on page 1067 | Version 10.1 |
| "WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97 - Return a list of activities" on page 1597 | Version 10.1 | "MON_GET_ACTIVITY table function - Return a list of activities" on page 514 | Version 10.1 |
| "WLM_GET_WORKLOAD_STATS_V97 - return workload statistics" on page 1601 | Version 10.1 | "WLM_GET_WORKLOAD_STATS table function - Return workload statistics" on page 1069 | Version 10.1 |
| "DB_PARTITIONS" on page 1180 | Version 9.8 | "DB2_MEMBER and DB2_CF administrative views and DB2_GET_INSTANCE_INFO table function" on page 376 | Version 9.8 |
| "SNAPDB_MEMORY_POOL administrative view and SNAP_GET_DB_MEMORY_POOL table function - Retrieve database level memory usage information" on page 1320 | Version 9.7 | "MON_GET_MEMORY_SET - get memory set information" on page 682
"MON_GET_MEMORY_POOL - get memory pool information" on page 679 | Version 9.7 Fix Pack 5 |
| "SNAPDBM_MEMORY_POOL administrative view and SNAP_GET_DBM_MEMORY_POOL table function - Retrieve database manager level memory usage information" on page 1339 | Version 9.7 | "MON_GET_MEMORY_SET - get memory set information" on page 682
"MON_GET_MEMORY_POOL - get memory pool information" on page 679 | Version 9.7 Fix Pack 5 |
| "LOCKS_HELD administrative view - Retrieve information about locks held" on page 1236 | Version 9.7 | "MON_GET_APPL_LOCKWAIT - Get information about locks for which an application is waiting" on page 561
"MON_GET_LOCKS - List all locks in the currently connected database" on page 674
"MON_FORMAT_LOCK_NAME - Format the internal lock name and return details" on page 483
"MON_LOCKWAITS administrative view - Retrieve metrics for applications that are waiting to obtain locks" on page 441 | Version 9.7 Fix Pack 1 |
| "LOCKWAITS administrative view - Retrieve current lockwaits information" on page 1239 | Version 9.7 | "MON_GET_APPL_LOCKWAIT - Get information about locks for which an application is waiting" on page 561
"MON_GET_LOCKS - List all locks in the currently connected database" on page 674
"MON_FORMAT_LOCK_NAME - Format the internal lock name and return details" on page 483
"MON_LOCKWAITS administrative view - Retrieve metrics for applications that are waiting to obtain locks" on page 441 | Version 9.7 Fix Pack 1 |
| "SNAPLOCK administrative view and SNAP_GET_LOCK table function - Retrieve lock logical data group snapshot information" on page 1365 | Version 9.7 | "MON_GET_APPL_LOCKWAIT - Get information about locks for which an application is waiting" on page 561
"MON_GET_LOCKS - List all locks in the currently connected database" on page 674
"MON_FORMAT_LOCK_NAME - Format the internal lock name and return details" on page 483
"MON_LOCKWAITS administrative view - Retrieve metrics for applications that are waiting to obtain locks" on page 441 | Version 9.7 Fix Pack 1 |
| "SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function - Retrieve lockwait logical data group snapshot information" on page 1371 | Version 9.7 | "MON_GET_APPL_LOCKWAIT - Get information about locks for which an application is waiting" on page 561
"MON_GET_LOCKS - List all locks in the currently connected database" on page 674
"MON_FORMAT_LOCK_NAME - Format the internal lock name and return details" on page 483
"MON_LOCKWAITS administrative view - Retrieve metrics for applications that are waiting to obtain locks" on page 441 | Version 9.7 Fix Pack 1 |

Table 226. Deprecated SQL administrative routines or views and their replacement routines or views (continued)

| Deprecated routine or view | Deprecated since | New routine or view | Replacement available since |
|---|------------------|---|-----------------------------|
| "WLM_GET_ACTIVITY_DETAILS - Return detailed information about a specific activity" on page 1575 | Version 9.5 | "MON_GET_ACTIVITY_DETAILS table function - Return information about an activity as an XML document" on page 533 | Version 9.7 |
| "GET_DB_CONFIG" on page 1188 | Version 9.1 | "DBCFCG administrative view and DB_GET_CFG table function - Retrieve database configuration parameter information" on page 363 | Version 9.1 |
| "GET_DBM_CONFIG" on page 1190 | Version 9.1 | "DBMCFG administrative view and DBM_GET_CFG table function - Retrieve database manager configuration parameter information" on page 367 | Version 9.1 |
| GET_ROUTINE_SAR procedure | Version 9.1 | CREATE PROCEDURE | Version 9.1 |
| PUT_ROUTINE_SAR procedure | Version 9.1 | CREATE PROCEDURE | Version 9.1 |
| "SNAP_GET_STO_PATHS" on page 1388 | Version 9.1 | "ADMIN_GET_STORAGE_PATHS table function - retrieve automatic storage path information" on page 217 | Version 10.1 |
| "SYSINSTALLROUTINES" on page 1572 | Version 9.1 | Not available | |

The health monitor has been deprecated. The health monitor routines are also deprecated and might be removed from future releases. A new suite of GUI tools for managing data and data-centric applications is available and can be used instead of the discontinued Control Center tools. For more information, see Database management and application development tools.

The following section is a list of the deprecated health monitor routines:

- "HEALTH_CONT_HI" on page 1191
- "HEALTH_CONT_HI_HIS" on page 1193
- "HEALTH_CONT_INFO" on page 1195
- "HEALTH_DB_HI" on page 1197
- "HEALTH_DB_HI_HIS" on page 1200
- "HEALTH_DB_HIC" on page 1204
- "HEALTH_DB_HIC_HIS" on page 1206
- "HEALTH_DB_INFO" on page 1209
- "HEALTH_DBM_HI" on page 1211
- "HEALTH_DBM_HI_HIS" on page 1212
- "HEALTH_DBM_INFO" on page 1215
- "HEALTH_GET_ALERT_ACTION_CFG" on page 1216
- "HEALTH_GET_ALERT_CFG" on page 1220
- "HEALTH_GET_IND_DEFINITION" on page 1223
- "HEALTH_HI_REC" on page 1225
- "HEALTH_TBS_HI" on page 1227
- "HEALTH_TBS_HI_HIS" on page 1230
- "HEALTH_TBS_INFO" on page 1234

ADMIN_GET_DBP_MEM_USAGE table function - Get total memory consumption for instance

The ADMIN_GET_DBP_MEM_USAGE table function gets the total memory consumption for a given instance.

Note: This table function has been deprecated and replaced by the “ADMIN_GET_MEM_USAGE table function - Get total memory consumption for instance” on page 214.

The *ADMIN_GET_DBP_MEM_USAGE* table function takes an optional input argument *member* (INTEGER type), which specifies a valid database member number, and returns only statistics for that single database member. If the argument is omitted, statistics are returned for all active database members. When in a multi-member environment, if you specify -1 or a NULL value for *member*, data is returned from the currently connected member.

Syntax

```
►►—ADMIN_GET_DBP_MEM_USAGE—(—member—)—►►
```

The schema is SYSPROC.

Table function parameters

member

An optional input argument of type integer that specifies the database member from which the memory usage statistics will be retrieved. If -1 or the NULL value is specified, data will be returned from the currently connected member.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Information returned

Table 227. The result set for *ADMIN_GET_DBP_MEM_USAGE*

| Column Name | Data Type | Description |
|-----------------------|-----------|--|
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MAX_PARTITION_MEM | BIGINT | The maximum amount of instance memory (in bytes) allowed to be consumed in the database partition if an instance memory limit is enforced. |
| CURRENT_PARTITION_MEM | BIGINT | The amount of instance memory (in bytes) currently consumed in the database partition. |
| PEAK_PARTITION_MEM | BIGINT | The peak or high watermark consumption of instance memory (in bytes) in the database partition. |

Examples

Example 1: Retrieve memory usage statistics from database partition 3

```
SELECT * FROM TABLE (SYSPROC.ADMIN_GET_DBP_MEM_USAGE(3)) AS T
```

| DBPARTITIONNUM | MAX_PARTITION_MEM | CURRENT_PARTITION_MEM | PEAK_PARTITION_MEM |
|----------------|-------------------|-----------------------|--------------------|
| 3 | 500000000 | 381000000 | 481000000 |

1 record(s) selected.

Example 2: Retrieve memory usage statistics from the currently connected member (assuming the user is connected to the database at member 2.)

```
SELECT * FROM TABLE (SYSPROC.ADMIN_GET_DBP_MEM_USAGE(-1)) AS T
```

| DBPARTITIONNUM | MAX_PARTITION_MEM | CURRENT_PARTITION_MEM | PEAK_PARTITION_MEM |
|----------------|-------------------|-----------------------|--------------------|
| 2 | 500000000 | 381000000 | 481000000 |

1 record(s) selected.

Example 3: Retrieve memory usage statistics from all members

```
SELECT * FROM TABLE (SYSPROC.ADMIN_GET_DBP_MEM_USAGE()) AS T
```

| DBPARTITIONNUM | MAX_PARTITION_MEM | CURRENT_PARTITION_MEM | PEAK_PARTITION_MEM |
|----------------|-------------------|-----------------------|--------------------|
| 0 | 500000000 | 381000000 | 481000000 |
| 1 | 500000000 | 381000000 | 481000000 |
| 2 | 500000000 | 381000000 | 481000000 |
| 3 | 500000000 | 381000000 | 481000000 |

4 record(s) selected.

Example 4: Retrieve memory usage statistics in megabyte (MB) values

```
SELECT DBPARTITIONNUM, MAX_PARTITION_MEM/1048576 AS MAX_MEM_MB,  
CURRENT_PARTITION_MEM/1048576 AS CURRENT_MEM_MB, PEAK_PARTITION_MEM/1048576  
AS PEAK_MEM_MB FROM TABLE (SYSPROC.ADMIN_GET_DBP_MEM_USAGE()) AS T
```

| DBPARTITIONNUM | MAX_MEM_MB | CURRENT_MEM_MB | PEAK_MEM_MB |
|----------------|------------|----------------|-------------|
| 0 | 4590 | 1107 | 1107 |
| 1 | 4590 | 1108 | 1108 |
| 2 | 4590 | 1106 | 1106 |

3 record(s) selected.

ADMINTABCOMPRESSINFO administrative view and ADMIN_GET_TAB_COMPRESS_INFO table function (deprecated) - returns compressed information

The ADMINTABCOMPRESSINFO administrative view and the ADMIN_GET_TAB_COMPRESS_INFO table function return compression information for tables, materialized query tables (MQT) and hierarchy tables.

Note: This administrative view and the associated table function has been deprecated and replaced by the following functions:

- “ADMIN_GET_TAB_COMPRESS_INFO table function - estimate compression savings” on page 219. Note that this replacement function has the same name as the deprecated function, but takes fewer arguments than the deprecated function.

- “ADMIN_GET_TAB_DICTIONARY_INFO table function - report properties of existing table dictionaries” on page 221.

This topic consists of the following sections:

- “ADMINTABCOMPRESSINFO administrative view”
- “ADMIN_GET_TAB_COMPRESS_INFO table function” on page 1128

ADMINTABCOMPRESSINFO administrative view

The ADMINTABCOMPRESSINFO administrative view returns compression information for tables, materialized query tables (MQT) and hierarchy tables only. These table types are reported as T for table, S for materialized query tables and H for hierarchy tables in the SYSCAT.TABLES catalog view. The information is returned at both the data partition level and the database partition level for a table.

The schema is SYSIBMADM.

Refer to the Information returned by ADMINTABCOMPRESSINFO administrative view and ADMIN_GET_TAB_COMPRESS_INFO table for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- SELECT privilege on the ADMINTABCOMPRESSINFO administrative view
- CONTROL privilege on the ADMINTABCOMPRESSINFO administrative view
- DATAACCESS authority

In addition, one of the following privileges or authorities is also required:

- EXECUTE privilege on the ADMIN_GET_TAB_COMPRESS_INFO table function
- DATAACCESS authority

Default PUBLIC privilege

None

Examples

Example 1: Retrieve all compression information for all tables

```
SELECT * FROM SYSIBMADM.ADMINTABCOMPRESSINFO
```

The following is an example of output from this query:

| TABSCHEMA | TABNAME | DBPARTITIONNUM | DATA_PARTITION_ID | COMPRESS_ATTR | ... |
|-----------|------------|----------------|-------------------|---------------|-----|
| ... | ... | ... | ... | ... | ... |
| SYSIBM | SYSTABLES | 0 | 0 | N | ... |
| SYSIBM | SYSCOLUMNS | 0 | 0 | N | ... |
| ... | ... | ... | ... | ... | ... |
| SIMAP2 | STAFF | 0 | 0 | Y | ... |
| SIMAP2 | PARTTAB | 0 | 0 | Y | ... |
| ... | ... | ... | ... | ... | ... |

156 record(s) selected.

Output from this query (continued):

| ... | DICT_BUILDER | DICT_BUILD_TIMESTAMP | COMPRESS_DICT_SIZE | EXPAND_DICT_SIZE | ... |
|-----|--------------|----------------------|--------------------|------------------|-----|
| ... | ... | ... | ... | ... | ... |
| ... | NOT BUILT | - | 0 | 0 | ... |

```

... NOT BUILT      -                      0          0 ...
...
... REORG          2006-08-27-19.07.36.000000      13312      5312 ...
... REORG          2006-08-27-22.07.17.000000      5760       4248 ...
...

```

Output from this query (continued):

```

... ROWS_SAMPLED  PAGES_SAVED_PERCENT  BYTES_SAVED_PERCENT  AVG_COMPRESS_REC_LENGTH
... -----
...              0                  0                  0                  0
...              0                  0                  0                  0
...
...             35                  65                  84                 100
...             45                  76                  79                 98
...

```

Example 2: Determine the dictionary building action and time of dictionary creation for all tables.

```

SELECT TABSCHEMA, TABNAME, DBPARTITIONNUM, DATA_PARTITION_ID, DICT_BUILDER, DICT_BUILD_TIMESTAMP
FROM SYSIBMADM.ADMINTABCOMPRESSINFO

```

The following is an example of output from this query:

```

TABSCHEMA TABNAME      DBPARTITIONNUM  DATA_PARTITION_ID  DICT_BUILDER  DICT_BUILD_TIMESTAMP
-----
SYSIBM     SYSTABLES         0                0 NOT BUILT    -
SYSIBM     SYSCOLUMNS         0                0 NOT BUILT    -
...
SIMAP2     STAFF                0                0 REORG        2006-08-27-19.07.36.000000
SIMAP2     SALES                0                0 NOT BUILT    -
SIMAP2     CATALOG              0                0 NOT BUILT    -
...

```

156 record(s) selected.

ADMIN_GET_TAB_COMPRESS_INFO table function

The ADMIN_GET_TAB_COMPRESS_INFO table function returns the same information as the ADMINTABCOMPRESSINFO administrative view, but allows you to specify a schema, table name and an execution mode.

Refer to the Information returned by ADMINTABCOMPRESSINFO administrative view and ADMIN_GET_TAB_COMPRESS_INFO table for a complete list of information that can be returned.

Syntax

```

▶▶ ADMIN_GET_TAB_COMPRESS_INFO (—tabschema—, —tablename—, —execmode—) ▶▶

```

The schema is SYSPROC.

Table function parameters

tabschema

An input argument of type VARCHAR(128) that specifies a schema name.

tablename

An input argument of type VARCHAR(128) that specifies a table name, a materialized query table name or a hierarchy table name.

execmode

An input argument of type VARCHAR(30) that specifies the execution mode. The execution mode can be one of the following modes:

- 'REPORT'-- Reports compression information as of last generation. This is the default value.
- 'ESTIMATE'-- Generates new compression information based on the current table.

Authorization

EXECUTE privilege on the ADMIN_GET_TAB_COMPRESS_INFO function.

Default PUBLIC privilege

None

Examples

Example 1: Retrieve existing compression information for table SIMAP2.STAFF

```
SELECT * FROM TABLE (SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO('SIMAP2', 'STAFF', 'REPORT'))
AS T
```

The following is an example from output of this query:

| TABSCHEMA | TABNAME | DBPARTITIONNUM | DATA_PARTITION_ID | COMPRESS_ATTR | ... |
|-----------|---------|----------------|-------------------|---------------|-----|
| SIMAP2 | STAFF | 0 | 0 | Y | ... |

1 record(s) selected.

Output from this query (continued):

| ... | DICT_BUILDER | DICT_BUILD_TIMESTAMP | COMPRESS_DICT_SIZE | EXPAND_DICT_SIZE | ... |
|-----|--------------|----------------------------|--------------------|------------------|-----|
| ... | REORG | 2006-08-27-19.07.36.000000 | 13312 | 5312 | ... |

Output from this query (continued):

| ... | ROWS_SAMPLED | PAGES_SAVED_PERCENT | BYTES_SAVED_PERCENT | AVG_COMPRESS_REC_LENGTH | ... |
|-----|--------------|---------------------|---------------------|-------------------------|-----|
| ... | 35 | 65 | 84 | 100 | ... |

Example 2: Retrieve estimated compression information for table SIMAP2.STAFF as of now.

```
SELECT * FROM TABLE (SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO('SIMAP2', 'STAFF', 'ESTIMATE'))
AS T
```

The following is an example from output of this query:

| TABSCHEMA | TABNAME | DBPARTITIONNUM | DATA_PARTITION_ID | COMPRESS_ATTR | DICT_BUILDER | ... |
|-----------|---------|----------------|-------------------|---------------|----------------|-----|
| SIMAP2 | STAFF | 0 | 0 | Y | TABLE FUNCTION | ... |

1 record(s) selected.

Output from this query (continued):

| ... | DICT_BUILD_TIMESTAMP | COMPRESS_DICT_SIZE | EXPAND_DICT_SIZE | ROWS_SAMPLED | ... |
|-----|----------------------------|--------------------|------------------|--------------|-----|
| ... | 2006-08-28-19.18.13.000000 | 13508 | 6314 | 68 | ... |

Output from this query (continued):

| ... | PAGES_SAVED_PERCENT | BYTES_SAVED_PERCENT | AVG_COMPRESS_REC_LENGTH | ... |
|-----|---------------------|---------------------|-------------------------|-----|
| ... | 72 | 89 | 98 | ... |

Example 3: Determine the total dictionary size for all tables in the schema SIMAP2

```

SELECT TABSCHEMA, TABNAME, DICT_BUILDER,
       (COMPRESS_DICT_SIZE+EXPAND_DICT_SIZE) AS TOTAL_DICT_SIZE,
       DBPARTITIONNUM, DATA_PARTITION_ID
FROM TABLE (SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO('SIMAP2', '', 'REPORT')) AS T

```

Output from this query:

| TABSCHEMA | TABNAME | DICT_BUILDER | TOTAL_DICT_SIZE | DBPARTITIONNUM | DATA_PARTITION_ID |
|-----------|-----------|--------------|-----------------|----------------|-------------------|
| SIMAP2 | ACT | NOT BUILT | 0 | 0 | 0 |
| SIMAP2 | ADEFUSR | NOT BUILT | 0 | 0 | 0 |
| ... | | | | | |
| SIMAP2 | INVENTORY | NOT BUILT | 0 | 0 | 0 |
| SIMAP2 | ORG | NOT BUILT | 0 | 0 | 0 |
| SIMAP2 | PARTTAB | REORG | 10008 | 0 | 0 |
| SIMAP2 | PARTTAB | REORG | 5464 | 0 | 1 |
| SIMAP2 | PARTTAB | REORG | 8456 | 0 | 2 |
| SIMAP2 | PARTTAB | REORG | 6960 | 0 | 3 |
| SIMAP2 | PARTTAB | REORG | 7136 | 0 | 4 |
| ... | | | | | |
| SIMAP2 | STAFF | REORG | 18624 | 0 | 0 |
| SIMAP2 | SUPPLIERS | NOT BUILT | 0 | 0 | 0 |
| SIMAP2 | TESTTABLE | NOT BUILT | 0 | 0 | 0 |

28 record(s) selected.

Example 4: View a report of the dictionary information of tables in the SIMAP2 schema.

```

SELECT * FROM TABLE (SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO('SIMAP2', '', 'REPORT'))
AS T

```

Output from this query:

| TABSCHEMA | TABNAME | DBPARTITIONNUM | DATA_PARTITION_ID | COMPRESS_ATTR | DICT_BUILDER | ... |
|-----------|---------|----------------|-------------------|---------------|--------------|-----|
| SIMAP2 | T1 | 0 | 0 | Y | NOT BUILT | ... |
| SIMAP2 | T2 | 0 | 0 | N | REORG | ... |
| SIMAP2 | T3 | 0 | 0 | Y | INSPECT | ... |
| SIMAP2 | T4 | 0 | 0 | N | NOT BUILT | ... |

4 record(s) selected.

Output from this query (continued):

| ... | DICT_BUILD_TIMESTAMP | COMPRESS_DICT_SIZE | EXPAND_DICT_SIZE | ROWS_SAMPLED | ... |
|-----|----------------------------|--------------------|------------------|--------------|-----|
| ... | - | 0 | 0 | 0 | ... |
| ... | 2007-02-03-17.35.28.000000 | 1280 | 2562 | - | ... |
| ... | 2007-02-03-17.35.44.000000 | 1340 | 2232 | - | ... |
| ... | - | 0 | 0 | 0 | ... |

Output from this query (continued):

| ... | PAGES_SAVED_PERCENT | BYTES_SAVED_PERCENT | AVG_COMPRESS_REC_LENGTH | ... |
|-----|---------------------|---------------------|-------------------------|-----|
| ... | 0 | 0 | 0 | ... |
| ... | - | - | - | ... |
| ... | - | - | - | ... |
| ... | 0 | 0 | 0 | ... |

Usage notes

- If both the *tabschema* and *tabname* are specified, information is returned for that specific table only.
- If the *tabschema* is specified but *tabname* is empty (") or NULL, information is returned for all tables in the given schema.
- If the *tabschema* is empty (") or NULL and *tabname* is specified, an error is returned. To retrieve information for a specific table, the table must be identified by both schema and table name.
- If both *tabschema* and *tabname* are empty (") or NULL, information is returned for all tables.

- If *tabschema* or *tabname* do not exist, or *tabname* does not correspond to a table name (type T), a materialized query table name (type S) or a hierarchy table name (type H), an empty result set is returned.
- When the ADMIN_GET_TAB_COMPRESS_INFO table function is retrieving data for a given table, it will acquire a shared lock on the corresponding row of SYSTABLES to ensure consistency of the data that is returned (for example, to ensure that the table is not altered while information is being retrieved for it). The lock will only be held for as long as it takes to retrieve the compression information for the table, and not for the duration of the table function call.
- If the queried table is a non-XML table, there will be a row returned for the XML storage object (XDA).

Information returned

Table 228. Information returned by ADMIN_TABCOMPRESSINFO administrative view and ADMIN_GET_TAB_COMPRESS_INFO

| Column Name | Data Type | Description |
|----------------------|--------------|---|
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name monitor element |
| TABNAME | VARCHAR(128) | table_name - Table name monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data partition identifier monitor element |
| COMPRESS_ATTR | CHAR(1) | The state of the COMPRESS attribute on the table which can be one of the following values: <ul style="list-style-type: none"> • 'Y' = Row compression is set to yes • 'N' = Row compression is set to no |
| DICT_BUILDER | VARCHAR(30) | Code path taken to build the dictionary which can be one of the following values: <ul style="list-style-type: none"> • 'INSPECT' = INSPECT ROWCOMPESTIMATE • 'LOAD' = LOAD INSERT/REPLACE • 'NOT BUILT' = no dictionary available • 'REDISTRIBUTE' = REDISTRIBUTE • 'REORG' = REORG RESETDICTIONARY • 'TABLE GROWTH' = INSERT • 'TABLE FUNCTION' = built by table function for the 'ESTIMATE' option |
| DICT_BUILD_TIMESTAMP | TIMESTAMP | Timestamp of when the dictionary was built. Timestamp granularity is to the second. If no dictionary is available, then the timestamp is NULL. |
| COMPRESS_DICT_SIZE | BIGINT | Size of compression dictionary measured in bytes. |
| EXPAND_DICT_SIZE | BIGINT | Size of the expansion dictionary measured in bytes. If a historical dictionary exists, this value is the sum of the current and historical dictionary sizes. |
| ROWS_SAMPLED | BIGINT | Number of records that contributed to building the dictionary. Migrated tables with compression dictionaries will return NULL in this column. |
| PAGES_SAVED_PERCENT | SMALLINT | Percentage of pages saved from compression. This information is based on the record data in the sample buffer only. Migrated tables with compression dictionaries will return NULL in this column. |

Table 228. Information returned by ADMIN_TABCOMPRESSINFO administrative view and ADMIN_GET_TAB_COMPRESS_INFO (continued)

| Column Name | Data Type | Description |
|-------------------------|-----------|--|
| BYTES_SAVED_PERCENT | SMALLINT | Percentage of bytes saved from compression. This information is based on the record data in the sample buffer only. Migrated tables with compression dictionaries will return NULL in this column. |
| AVG_COMPRESS_REC_LENGTH | SMALLINT | Average compressed record length of the records contributing to building the dictionary. Migrated tables with compression dictionaries will return NULL in this column. |

ADMIN_GET_TAB_COMPRESS_INFO_V97

The ADMIN_GET_TAB_COMPRESS_INFO_V97 table function returns compression information for tables, materialized query tables (MQT) and hierarchy tables.

Note: This table function has been deprecated and replaced by the ADMIN_GET_TAB_COMPRESS_INFO table function and the ADMIN_GET_TAB_DICTIONARY_INFO table function.

Refer to the Table 229 on page 1137 table for a complete list of information that can be returned.

Syntax

►►—ADMIN_GET_TAB_COMPRESS_INFO_V97—(—*tabschema*—,—*tabname*—,—*execmode*—)————►►

The schema is SYSPROC.

Table function parameters

tabschema

An input argument of type VARCHAR(128) that specifies a schema name.

tabname

An input argument of type VARCHAR(128) that specifies a table name, a materialized query table name or a hierarchy table name.

execmode

An input argument of type VARCHAR(30) that specifies the execution mode. The execution mode can be one of the following modes:

- 'REPORT' -- Reports compression information as of last generation. This is the default value.
- 'ESTIMATE' -- Generates new compression information based on the current table.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority

- SQLADM authority

Default PUBLIC privilege

None

Examples

Example 1: Retrieve existing compression information for table SIMAP2.STAFF

```
SELECT *
FROM TABLE (
  SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO_V97('SIMAP2', 'STAFF', 'REPORT'))
AS T
```

The following is an example from output of this query:

| TABSCHEMA | TABNAME | DBPARTITIONNUM | DATA_PARTITION_ID | COMPRESS_ATTR | DICT_BUILDER | ... |
|-----------|---------|----------------|-------------------|---------------|--------------|-----|
| SIMAP2 | STAFF | 0 | 4 | Y | REORG | ... |
| SIMAP2 | STAFF | 0 | 4 | Y | NOT BUILT | ... |

2 record(s) selected.

Output from this query (continued):

| DICT_BUILD_TIMESTAMP | COMPRESS_DICT_SIZE | EXPAND_DICT_SIZE | ROWS_SAMPLED | ... |
|----------------------------|--------------------|------------------|--------------|-----|
| 2009-03-31-12.19.30.000000 | 13312 | 5296 | 35 | ... |
| - | 0 | 0 | 0 | ... |

Output from this query (continued):

| PAGES_SAVED_PERCENT | BYTES_SAVED_PERCENT | AVG_COMPRESS_REC_LENGTH | OBJECT_TYPE |
|---------------------|---------------------|-------------------------|-------------|
| 38 | 38 | 27 | DATA |
| 0 | 0 | 0 | XML |

Example 2: Retrieve estimated compression information for table SIMAP2.STAFF as of now.

```
SELECT *
FROM TABLE (
  SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO_V97('SIMAP2', 'STAFF', 'ESTIMATE'))
AS T
```

The following is an example from output of this query:

| TABSCHEMA | TABNAME | DBPARTITIONNUM | DATA_PARTITION_ID | COMPRESS_ATTR | DICT_BUILDER | ... |
|-----------|---------|----------------|-------------------|---------------|----------------|-----|
| SIMAP2 | STAFF | 0 | 4 | Y | TABLE FUNCTION | ... |
| SIMAP2 | STAFF | 0 | 4 | Y | TABLE FUNCTION | ... |

2 record(s) selected.

Output from this query (continued):

| DICT_BUILD_TIMESTAMP | COMPRESS_DICT_SIZE | EXPAND_DICT_SIZE | ROWS_SAMPLED | ... |
|----------------------------|--------------------|------------------|--------------|-----|
| 2009-03-31-12.27.06.000000 | 13312 | 5296 | 35 | ... |
| 2009-03-31-12.27.06.000000 | 13312 | 9544 | 8 | ... |

Output from this query (continued):

| PAGES_SAVED_PERCENT | BYTES_SAVED_PERCENT | AVG_COMPRESS_REC_LENGTH | OBJECT_TYPE |
|---------------------|---------------------|-------------------------|-------------|
| 38 | 38 | 27 | DATA |
| 75 | 75 | 95 | XML |

Example 3: Determine the total dictionary size for all objects in tables in the schema SIMAP2

```
SELECT TABSCHEMA, TABNAME, OBJECT_TYPE, DICT_BUILDER, (
    COMPRESS_DICT_SIZE+EXPAND_DICT_SIZE)
    AS TOTAL_DICT_SIZE, DBPARTITIONNUM, DATA_PARTITION_ID
FROM TABLE (
    SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO_V97('SIMAP2', '', 'REPORT'))
AS T
```

Output from this query:

| TABSCHEMA | TABNAME | OBJECT_TYPE | DICT_BUILDER | ... |
|-----------|------------|-------------|--------------|-----|
| SIMAP2 | ACT | DATA | NOT BUILT | ... |
| SIMAP2 | ACT | XML | NOT BUILT | ... |
| SIMAP2 | ADEFUSR | DATA | INSPECT | ... |
| SIMAP2 | ADEFUSR | XML | NOT BUILT | ... |
| ... | | | | |
| SIMAP2 | CUSTOMER | DATA | REORG | ... |
| SIMAP2 | CUSTOMER | XML | REORG | ... |
| SIMAP2 | DEPARTMENT | DATA | NOT BUILT | ... |
| SIMAP2 | DEPARTMENT | XML | NOT BUILT | ... |
| ... | | | | |
| SIMAP2 | STAFF | DATA | REORG | ... |
| SIMAP2 | STAFF | XML | NOT BUILT | ... |
| SIMAP2 | SUPPLIERS | DATA | TABLE GROWTH | ... |
| SIMAP2 | SUPPLIERS | XML | NOT BUILT | ... |

44 record(s) selected.

Output from this query (continued):

| TOTAL_DICT_SIZE | DBPARTITIONNUM | DATA_PARTITION_ID |
|-----------------|----------------|-------------------|
| 0 | 0 | 0 |
| 0 | 0 | 0 |
| 1890 | 0 | 0 |
| 0 | 0 | 0 |
| ... | | |
| 6968 | 0 | 1 |
| 24256 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 0 |
| ... | | |
| 18608 | 0 | 4 |
| 0 | 0 | 4 |
| 6960 | 0 | 2 |
| 0 | 0 | 2 |

Example 4: View a report of the dictionary information of tables in the SIMAP2 schema.

```
SELECT * FROM TABLE (
    SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO_V97('SIMAP2', '', 'REPORT'))
AS T
```

Output from this query:

| TABSCHEMA | TABNAME | DBPARTITIONNUM | DATA_PARTITION_ID | COMPRESS_ATTR | DICT_BUILDER | ... |
|-----------|---------|----------------|-------------------|---------------|--------------|-----|
| SIMAP2 | ACT | 0 | 0 | N | NOT BUILT | ... |
| SIMAP2 | ACT | 0 | 0 | N | NOT BUILT | ... |

| | | | | | |
|--------|-----------|---|-----|-----------|-----|
| SIMAP2 | ADEFUSR | 0 | 0 N | INSPECT | ... |
| SIMAP2 | ADEFUSR | 0 | 0 N | NOT BUILT | ... |
| ... | | | | | |
| SIMAP2 | CUSTOMER | 0 | 1 Y | REORG | ... |
| SIMAP2 | CUSTOMER | 0 | 1 Y | REORG | ... |
| ... | | | | | |
| SIMAP2 | STAFF | 0 | 4 Y | REORG | ... |
| SIMAP2 | STAFF | 0 | 4 Y | NOT BUILT | ... |
| SIMAP2 | SUPPLIERS | 0 | 2 N | NOT BUILT | ... |
| SIMAP2 | SUPPLIERS | 0 | 2 N | NOT BUILT | ... |

44 record(s) selected.

Output from this query (continued):

| DICT_BUILD_TIMESTAMP | COMPRESS_DICT_SIZE | EXPAND_DICT_SIZE | ROWS_SAMPLED | ... |
|----------------------------|--------------------|------------------|--------------|-----|
| - | 0 | 0 | 0 | ... |
| - | 0 | 0 | 0 | ... |
| 2009-03-31-12.11.02.000000 | 290 | 1890 | 22 | ... |
| - | 0 | 0 | 0 | ... |
| ... | | | | |
| 2009-03-31-11.08.18.000000 | 3968 | 3000 | 6 | ... |
| 2009-03-31-11.08.18.000000 | 13312 | 10944 | 6 | ... |
| ... | | | | |
| 2009-03-31-12.19.30.000000 | 13312 | 5296 | 35 | ... |
| - | 0 | 0 | 0 | ... |
| - | 0 | 0 | 0 | ... |
| - | 0 | 0 | 0 | ... |

Output from this query (continued):

| PAGES_SAVED_PERCENT | BYTES_SAVED_PERCENT | AVG_COMPRESS_REC_LENGTH | OBJECT_TYPE |
|---------------------|---------------------|-------------------------|-------------|
| 0 | 0 | 0 | DATA |
| 0 | 0 | 0 | XML |
| 20 | 25 | 21 | DATA |
| 0 | 0 | 0 | XML |
| ... | | | |
| 70 | 70 | 31 | DATA |
| 66 | 66 | 235 | XML |
| ... | | | |
| 38 | 38 | 27 | DATA |
| 0 | 0 | 0 | XML |
| 0 | 0 | 0 | DATA |
| 0 | 0 | 0 | XML |

Example 5: View a report of the dictionary information of DATA objects of tables in the SIMAP2 schema.

```
SELECT * FROM TABLE (
    SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO_V97('SIMAP2',' ','REPORT'))
WHERE OBJECT_TYPE='DATA'
```

Output from this query:

| TABSCHEMA | TABNAME | DBPARTITIONNUM | DATA_PARTITION_ID | COMPRESS_ATTR | DICT_BUILDER | ... |
|-----------|------------|----------------|-------------------|---------------|--------------|-----|
| SIMAP2 | ACT | 0 | 0 N | NOT BUILT | ... | |
| SIMAP2 | ADEFUSR | 0 | 0 N | INSPECT | ... | |
| ... | | | | | | |
| SIMAP2 | CUSTOMER | 0 | 1 Y | REORG | ... | |
| SIMAP2 | DEPARTMENT | 1 | 0 N | NOT BUILT | ... | |
| ... | | | | | | |
| SIMAP2 | STAFF | 0 | 4 Y | REORG | ... | |
| SIMAP2 | SUPPLIERS | 0 | 2 N | NOT BUILT | ... | |

22 record(s) selected.

Output from this query (continued):

| DICT_BUILD_TIMESTAMP | COMPRESS_DICT_SIZE | EXPAND_DICT_SIZE | ROWS_SAMPLED... |
|----------------------------|--------------------|------------------|-----------------|
| - | 0 | 0 | 0 ... |
| 2009-03-31-12.11.02.000000 | 290 | 1890 | 22 ... |
| ... | | | |
| 2009-03-31-11.08.18.000000 | 3968 | 3000 | 6 ... |
| - | 0 | 0 | 0 ... |
| ... | | | |
| 2009-03-31-12.19.30.000000 | 13312 | 5296 | 35 ... |
| - | 0 | 0 | 0 ... |

Output from this query (continued):

| PAGES_SAVED_PERCENT | BYTES_SAVED_PERCENT | AVG_COMPRESS_REC_LENGTH | OBJECT_TYPE |
|---------------------|---------------------|-------------------------|-------------|
| 0 | 0 | 0 | DATA |
| 20 | 25 | 21 | DATA |
| 70 | 70 | 31 | DATA |
| 0 | 0 | 0 | DATA |
| 38 | 38 | 27 | DATA |
| 0 | 0 | 0 | DATA |

Example 6: View a report of the dictionary information of XML objects of the CUSTOMER table in the SIMAP2 schema.

```
SELECT * FROM TABLE (
    SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO_V97('SIMAP2', 'CUSTOMER', 'REPORT'))
WHERE OBJECT_TYPE='XML'
```

Output from this query:

| TABSCHEMA | TABNAME | DBPARTITIONNUM | DATA_PARTITION_ID | COMPRESS_ATTR | DICT_BUILDER | ... |
|-----------|----------|----------------|-------------------|---------------|--------------|-----|
| SIMAP2 | CUSTOMER | 0 | 1 | Y | REORG | ... |

Output from this query (continued):

| DICT_BUILD_TIMESTAMP | COMPRESS_DICT_SIZE | EXPAND_DICT_SIZE | ROWS_SAMPLED | ... |
|----------------------------|--------------------|------------------|--------------|-----|
| 2009-03-31-11.08.18.000000 | 13312 | 10944 | 6 | ... |

Output from this query (continued):

| PAGES_SAVED_PERCENT | BYTES_SAVED_PERCENT | AVG_COMPRESS_REC_LENGTH | OBJECT_TYPE |
|---------------------|---------------------|-------------------------|-------------|
| 66 | 66 | 235 | XML |

Usage notes

- If both the *tabschema* and *tabname* are specified, information is returned for that specific table only.
- If the *tabschema* is specified but *tabname* is empty (") or NULL, information is returned for all tables in the given schema.
- If the *tabschema* is empty (") or NULL and *tabname* is specified, an error is returned. To retrieve information for a specific table, the table must be identified by both schema and table name.
- If both *tabschema* and *tabname* are empty (") or NULL, information is returned for all tables.
- If *tabschema* or *tabname* do not exist, or *tabname* does not correspond to a table name (type T), a materialized query table name (type S) or a hierarchy table name (type H), an empty result set is returned.

- When the ADMIN_GET_TAB_COMPRESS_INFO_V97 table function is retrieving data for a given table, it will acquire a shared lock on the corresponding row of SYSTABLES to ensure consistency of the data that is returned (for example, to ensure that the table is not altered while information is being retrieved for it). The lock will only be held for as long as it takes to retrieve the compression information for the table, and not for the duration of the table function call.
- If the queried table is a non-XML table, there will be a row returned for the XML storage object (XDA).

Information returned

Table 229. Information returned by ADMIN_GET_TAB_COMPRESS_INFO_V97

| Column Name | Data Type | Description |
|----------------------|--------------|---|
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name monitor element |
| TABNAME | VARCHAR(128) | table_name - Table name monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data partition identifier monitor element |
| COMPRESS_ATTR | CHAR(1) | The state of the COMPRESS attribute on the table which can be one of the following values: <ul style="list-style-type: none"> • 'Y' = Row compression is set to yes • 'N' = Row compression is set to no |
| DICT_BUILDER | VARCHAR(30) | Code path taken to build the dictionary which can be one of the following values: <ul style="list-style-type: none"> • 'INSPECT' = INSPECT ROWCOMPESTIMATE • 'LOAD' = LOAD INSERT/REPLACE • 'NOT BUILT' = no dictionary available • 'REDISTRIBUTE' = REDISTRIBUTE • 'REORG' = REORG RESETDICTIONARY • 'TABLE GROWTH' = INSERT • 'TABLE FUNCTION' = built by table function for the 'ESTIMATE' option |
| DICT_BUILD_TIMESTAMP | TIMESTAMP | Timestamp of when the dictionary was built. Timestamp granularity is to the second. If no dictionary is available, then the timestamp is NULL. |
| COMPRESS_DICT_SIZE | BIGINT | Size of compression dictionary measured in bytes. |
| EXPAND_DICT_SIZE | BIGINT | Size of the expansion dictionary measured in bytes. If a historical dictionary exists, this value is the sum of the current and historical dictionary sizes. |
| ROWS_SAMPLED | BIGINT | Number of records that contributed to building the dictionary. Migrated tables with compression dictionaries will return NULL in this column. |
| PAGES_SAVED_PERCENT | SMALLINT | Percentage of pages saved from compression. This information is based on the record data in the sample buffer only. Migrated tables with compression dictionaries will return NULL in this column. |
| BYTES_SAVED_PERCENT | SMALLINT | Percentage of bytes saved from compression. This information is based on the record data in the sample buffer only. Migrated tables with compression dictionaries will return NULL in this column. |

Table 229. Information returned by ADMIN_GET_TAB_COMPRESS_INFO_V97 (continued)

| Column Name | Data Type | Description |
|-------------------------|------------|---|
| AVG_COMPRESS_REC_LENGTH | SMALLINT | Average compressed record length of the records contributing to building the dictionary. Migrated tables with compression dictionaries will return NULL in this column. |
| OBJECT_TYPE | VARCHAR(4) | objtype - Object type monitor element |

ADMIN_GET_TAB_INFO_V95 table function - Retrieve size and state information for tables

The ADMIN_GET_TAB_INFO_V95 table function provides methods to retrieve table size and state information that is not currently available in the catalog views.

Note: This table function has been deprecated and replaced by the “ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function - retrieve table size and state information” on page 224.

Refer to the Table 230 on page 1140 table for a complete list of information that can be returned.

Syntax

►►—ADMIN_GET_TAB_INFO_V95—(—*tabschema*—,—*tabname*—)————►►

The schema is SYSPROC.

Table function parameters

tabschema

An input argument of type VARCHAR(128) that specifies a schema name.

tabname

An input argument of type VARCHAR(128) that specifies a table name, a materialized query table name or a hierarchy table name.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Retrieve size and state information for the table DBUSER1.EMPLOYEE.

```
SELECT * FROM TABLE (SYSPROC.ADMIN_GET_TAB_INFO_V95('DBUSER1', 'EMPLOYEE'))
AS T
```

Example 2: Suppose there exists a non-partitioned table (DBUSER1.EMPLOYEE), with all associated objects (for example, indexes and LOBs) stored in a single table space. Calculate how much physical space the table is using in the table space:

```
SELECT (data_object_p_size + index_object_p_size + long_object_p_size +
lob_object_p_size + xml_object_p_size) as total_p_size
FROM TABLE( SYSPROC.ADMIN_GET_TAB_INFO_V95( 'DBUSER1', 'EMPLOYEE' )) AS T
```

Calculate how much space would be required if the table were moved to another table space, where the new table space has the same page size and extent size as the original table space:

```
SELECT (data_object_l_size + index_object_l_size + long_object_l_size +
lob_object_l_size + xml_object_l_size) as total_l_size
FROM TABLE( SYSPROC.ADMIN_GET_TAB_INFO_V95( 'DBUSER1', 'EMPLOYEE' )) AS T
```

Example 3: Check the current type of statistics information collected for table T1

```
db2 => select substr(tabschema, 1, 10) as tbschema, substr(tabname, 1, 10)
as tname, statstype from SYSIBMADM.ADMINTABINFO where tabname = 'T1';
```

| TBSHEMA | TBNAME | STATTYPE |
|----------|--------|----------|
| DB2USER1 | T1 | U |

1 record(s) selected.

Usage notes

- If both the *tabschema* and *tabname* are specified, information is returned for that specific table only.
- If the *tabschema* is specified but *tabname* is empty (") or NULL, information is returned for all tables in the given schema.
- If the *tabschema* is empty (") or NULL and *tabname* is specified, an error is returned. To retrieve information for a specific table, the table must be identified by both schema and table name.
- If both *tabschema* and *tabname* are empty (") or NULL, information is returned for all tables.
- If *tabschema* or *tabname* do not exist, or *tabname* does not correspond to a table name (type T), a materialized query table name (type S) or a hierarchy table name (type H), an empty result set is returned.
- When the ADMIN_GET_TAB_INFO_V95 table function is retrieving data for a given table, it will acquire a shared lock on the corresponding row of SYSTABLES to ensure consistency of the data that is returned (for example, to ensure that the table is not dropped while information is being retrieved for it). The lock will only be held for as long as it takes to retrieve the size and state information for the table, not for the duration of the table function call.
- Physical size reported for tables in SMS table spaces is the same as logical size.
- When an inplace reorg is active on a table, the physical size for the data object (DATA_OBJECT_P_SIZE) will not be calculated. Only the logical size will be returned. You can tell if an inplace reorg is active on the table by looking at the INPLACE_REORG_STATUS output column.

REDISTRIBUTING_PENDING

1. no redistribute has been run for the given table N

2. redistribute started to run on the database partition group but not on the table N
3. redistribute failed in the phase before moving data N
4. redistribute failed in the phase of moving data Y
5. redistribute completely successfully and committed for the table. N

Information returned

Table 230. Information returned by the ADMIN_GET_TAB_INFO_V95 table function

| Column name | Data type | Description |
|--------------------|--------------|--|
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name monitor element |
| TABNAME | VARCHAR(128) | table_name - Table name monitor element |
| TABTYPE | CHAR(1) | Table type: <ul style="list-style-type: none"> • 'H' = hierarchy table • 'S' = materialized query table • 'T' = table |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data partition identifier monitor element |
| AVAILABLE | CHAR(1) | State of the table: <ul style="list-style-type: none"> • 'N' = the table is unavailable. If the table is unavailable, all other output columns relating to the size and state will be NULL. • 'Y' = the table is available. Note: Rollforward through an unrecoverable load will put a table into the unavailable state. |
| DATA_OBJECT_L_SIZE | BIGINT | Data object logical size. Amount of disk space logically allocated for the table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated for the table (for example, in the case of a logical table truncation). For multi-dimensional clustering (MDC) tables, this size includes the logical size of the block map object. The size returned takes into account full extents that are logically allocated for the table and, for objects created in DMS table spaces, an estimate of the Extent Map Page (EMP) extents. This size represents the logical size of the base table only. Space consumed by LOB data, Long Data, Indexes and XML objects are reported by other columns. |
| DATA_OBJECT_P_SIZE | BIGINT | Data object physical size. Amount of disk space physically allocated for the table, reported in kilobytes. For MDC tables, this size includes the size of the block map object. The size returned takes into account full extents allocated for the table and includes the EMP extents for objects created in DMS table spaces. This size represents the physical size of the base table only. Space consumed by LOB data, Long Data, Indexes and XML objects are reported by other columns. |

Table 230. Information returned by the ADMIN_GET_TAB_INFO_V95 table function (continued)

| Column name | Data type | Description |
|---------------------|-----------|---|
| INDEX_OBJECT_L_SIZE | BIGINT | Index object logical size. Amount of disk space logically allocated for the indexes defined on the table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold index data for the table (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for the indexes and, for indexes created in DMS table spaces, an estimate of the EMP extents. This value is only reported for non-partitioned tables. For partitioned tables, this value will be 0. |
| INDEX_OBJECT_P_SIZE | BIGINT | Index object physical size. Amount of disk space physically allocated for the indexes defined on the table, reported in kilobytes. The size returned takes into account full extents allocated for the indexes and includes the EMP extents for indexes created in DMS table spaces. This value is only reported for non-partitioned tables. For partitioned tables this value will be 0. |
| LONG_OBJECT_L_SIZE | BIGINT | Long object logical size. Amount of disk space logically allocated for long field data in a table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold long field data for the table (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for long field data and, for long field data created in DMS table spaces, an estimate of the EMP extents. |
| LONG_OBJECT_P_SIZE | BIGINT | Long object physical size. Amount of disk space physically allocated for long field data in a table, reported in kilobytes. The size returned takes into account full extents allocated for long field data and includes the EMP extents for long field data created in DMS table spaces. |
| LOB_OBJECT_L_SIZE | BIGINT | LOB object logical size. Amount of disk space logically allocated for LOB data in a table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold LOB data for the table (for example, in the case of a logical table truncation). The size includes space logically allocated for the LOB allocation object. The size returned takes into account full extents that are logically allocated for LOB data and, for LOB data created in DMS table spaces, an estimate of the EMP extents. |
| LOB_OBJECT_P_SIZE | BIGINT | LOB object physical size. Amount of disk space physically allocated for LOB data in a table, reported in kilobytes. The size includes space allocated for the LOB allocation object. The size returned takes into account full extents allocated for LOB data and includes the EMP extents for LOB data created in DMS table spaces. |

Table 230. Information returned by the ADMIN_GET_TAB_INFO_V95 table function (continued)

| Column name | Data type | Description |
|----------------------|-------------|--|
| XML_OBJECT_L_SIZE | BIGINT | XML object logical size. Amount of disk space logically allocated for XML data in a table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold XML data for the table (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for XML data and, for XML data created in DMS table spaces, an estimate of the EMP extents. |
| XML_OBJECT_P_SIZE | BIGINT | XML object physical size. Amount of disk space physically allocated for XML data in a table, reported in kilobytes. The size returned takes into account full extents allocated for XML data and includes the EMP extents for XML data created in DMS table spaces. |
| INDEX_TYPE | SMALLINT | Indicates the type of indexes currently in use for the table. Returns: <ul style="list-style-type: none"> • 1 if type-1 indexes are being used. • 2 if type-2 indexes are being used. |
| REORG_PENDING | CHAR(1) | A value of 'Y' indicates that a reorg recommended alter has been applied to the table and a classic (offline) reorg is required. Otherwise 'N' is returned. |
| INPLACE_REORG_STATUS | VARCHAR(10) | Current status of an inplace table reorganization on the table. The status value can be one of the following values: <ul style="list-style-type: none"> • ABORTED (in a PAUSED state, but unable to RESUME; STOP is required) • EXECUTING • NULL (if no inplace reorg has been performed on the table) • PAUSED |
| LOAD_STATUS | VARCHAR(12) | Current status of a load operation against the table. The status value can be one of the following values: <ul style="list-style-type: none"> • IN_PROGRESS • NULL (if there is no load in progress for the table and the table is not in load pending state) • PENDING |
| READ_ACCESS_ONLY | CHAR(1) | 'Y' if the table is in Read Access Only state, 'N' otherwise. A value of 'N' should not be interpreted as meaning that the table is fully accessible. If a load is in progress or pending, a value of 'Y' means the table data is available for read access, and a value of 'N' means the table is inaccessible. Similarly, if the table status is set integrity pending (refer to SYSCAT.TABLES STATUS column), then a value of 'N' means the table is inaccessible. |
| NO_LOAD_RESTART | CHAR(1) | A value of 'Y' indicates the table is in a partially loaded state that will not allow a load restart. A value of 'N' is returned otherwise. |
| NUM_REORG_REC_ALTERS | SMALLINT | Number of reorg recommend alter operations (for example, alter operations after which a reorganization is required) that have been performed against this table since the last reorganization. |

Table 230. Information returned by the ADMIN_GET_TAB_INFO_V95 table function (continued)

| Column name | Data type | Description |
|-------------------------|-----------|---|
| INDEXES_REQUIRE_REBUILD | CHAR(1) | 'Y' if any of the indexes defined on the table require a rebuild, and 'N' otherwise. |
| LARGE_RIDS | CHAR(1) | Indicates whether or not the table is using large row IDs (RIDs) (4 byte page number, 2 byte slot number). A value of 'Y' indicates that the table is using large RIDs and 'N' indicates that it is not using large RIDs. A value of 'P' (pending) will be returned if the table supports large RIDs (that is, the table is in a large table space), but at least one of the indexes for the table has not been reorganized or rebuilt yet, so the table is still using 4 byte RIDs (which means that action must be taken to convert the table or indexes). |
| LARGE_SLOTS | CHAR(1) | Indicates whether or not the table is using large slots (which allows more than 255 rows per page). A value of 'Y' indicates that the table is using large slots and 'N' indicates that it is not using large slots. A value of 'P' (pending) will be returned if the table supports large slots (that is, the table is in a large table space), but there has been no offline table reorganization or table truncation operation performed on the table yet, so it is still using a maximum of 255 rows per page. |
| DICTIONARY_SIZE | BIGINT | Size of the dictionary, in bytes, used for row compression if a row compression dictionary exists for the table. |
| BLOCKS_PENDING_CLEANUP | BIGINT | blocks_pending_cleanup - Pending cleanup rolled-out blocks monitor element |
| STATSTYPE | CHAR(1) | <ul style="list-style-type: none"> 'F' = System fabricated statistics without table or index scan. These statistics are stored in memory and are different from what is stored in the system catalogs. This is a temporary state and eventually full statistics will be gathered by the database system and stored in the system catalogs. 'A' = System asynchronously gathered statistics. Statistics have been automatically collected by the database system by a background process and stored in the system catalogs. 'S' = System synchronously gathered statistics. Statistics have been automatically collected by the database system during SQL statement compilation. These statistics are stored in memory and are different from what is stored in the system catalogs. This is a temporary state and eventually the database system stores the statistics in the system catalogs. 'U' = User gathered statistics. Statistics gathering was initiated by the user through a utility such as RUNSTATS, CREATE INDEX, LOAD, REDISTRIBUTE or by manually updating system catalog statistics. NULL = unknown type |

ADMIN_GET_TAB_INFO_V97 table function - Retrieve size and state information for tables

The ADMIN_GET_TAB_INFO_V97 table function provides methods to retrieve table size and state information that is not currently available in the catalog views.

Note: This table function has been deprecated and replaced by the “ADMIN_TABINFO administrative view and ADMIN_GET_TAB_INFO table function - retrieve table size and state information” on page 224.

Refer to the Table 231 on page 1145 table for a complete list of information that can be returned.

Syntax

```
►►—ADMIN_GET_TAB_INFO_V97—(—tabschema—,—tabname—)————►►
```

The schema is SYSPROC.

Table function parameters

tabschema

An input argument of type VARCHAR(128) that specifies a schema name.

tabname

An input argument of type VARCHAR(128) that specifies a table name, a materialized query table name or a hierarchy table name.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Retrieve size and state information for the table DBUSER1.EMPLOYEE.

```
SELECT * FROM TABLE (SYSPROC.ADMIN_GET_TAB_INFO_V97('DBUSER1', 'EMPLOYEE'))
      AS T
```

Example 2: Suppose there exists a non-partitioned table (DBUSER1.EMPLOYEE), with all associated objects (for example, indexes and LOBs) stored in a single table space. Calculate how much physical space the table is using in the table space:

```
SELECT (data_object_p_size + index_object_p_size + long_object_p_size +
       lob_object_p_size + xml_object_p_size) as total_p_size
FROM TABLE( SYSPROC.ADMIN_GET_TAB_INFO_V97( 'DBUSER1', 'EMPLOYEE' )) AS T
```

Calculate how much space would be required if the table were moved to another table space, where the new table space has the same page size and extent size as the original table space:

```
SELECT (data_object_l_size + index_object_l_size + long_object_l_size +
       lob_object_l_size + xml_object_l_size) as total_l_size
FROM TABLE( SYSPROC.ADMIN_GET_TAB_INFO_V97( 'DBUSER1', 'EMPLOYEE' )) AS T
```

Example 3: Determine the total size for the compression dictionaries for the table DBUSER1.EMPLOYEE.

```
SELECT SUBSTR(TABSCHEMA,1,10) AS TBSchema, SUBSTR(TABNAME,1,10) AS TBNAME,
       DICTIONARY_SIZE + XML_DICTIONARY_SIZE AS TOTAL_DICTIONARY_SIZE
FROM TABLE(SYSPROC.ADMIN_GET_TAB_INFO_V97('DBUSER1','EMPLOYEE'))
```

Example 4: Determine the amount of space reclaimable from a multidimensional clustering table SAMPLE.STAFF:

```
SELECT RECLAIMABLE_SPACE
FROM TABLE(SYSPROC.ADMIN_GET_TAB_INFO_V97('SAMPLE','STAFF'))
```

Usage notes

- If both the *tabschema* and *tabname* are specified, information is returned for that specific table only.
- If the *tabschema* is specified but *tabname* is NULL or the empty string ("), then information is returned for all tables in the given schema.
- If the *tabschema* is NULL or the empty string (") and *tabname* is specified, then an error is returned. To retrieve information for a specific table, the table must be identified by both schema and table name.
- If both *tabschema* and *tabname* are NULL or the empty string ("), then information is returned for all tables.
- If *tabschema* or *tabname* do not exist, or *tabname* does not correspond to a table name (type T), a materialized query table name (type S) or a hierarchy table name (type H), an empty result set is returned.
- When the ADMIN_GET_TAB_INFO_V97 table function is retrieving data for a given table, it will acquire a shared lock on the corresponding row of SYSTABLES to ensure consistency of the data that is returned (for example, to ensure that the table is not dropped while information is being retrieved for it). The lock will only be held for as long as it takes to retrieve the size and state information for the table, not for the duration of the table function call.
- Physical size reported for tables in SMS table spaces is the same as logical size.
- When an inplace reorg is active on a table, the physical size for the data object (DATA_OBJECT_P_SIZE) will not be calculated. Only the logical size will be returned. You can tell if an inplace reorg is active on the table by looking at the INPLACE_REORG_STATUS output column.

REDISTRIBUTING_PENDING

1. no redistribute has been run for the given table N
2. redistribute started to run on the database partition group but not on the table N
3. redistribute failed in the phase before moving data N
4. redistribute failed in the phase of moving data Y
5. redistribute completely successfully and committed for the table N

Information returned

Table 231. Information returned by ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO_V97

| Column name | Data type | Description |
|-------------|--------------|--|
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name monitor element |
| TABNAME | VARCHAR(128) | table_name - Table name monitor element |

Table 231. Information returned by ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO_V97 (continued)

| Column name | Data type | Description |
|--------------------|-----------|--|
| TABTYPE | CHAR(1) | Table type: <ul style="list-style-type: none"> 'H' = hierarchy table 'S' = materialized query table 'T' = table |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data partition identifier monitor element |
| AVAILABLE | CHAR(1) | State of the table: <ul style="list-style-type: none"> 'N' = the table is unavailable. If the table is unavailable, all other output columns relating to the size and state will be NULL. 'Y' = the table is available. Note: Rollforward through an unrecoverable load will put a table into the unavailable state. |
| DATA_OBJECT_L_SIZE | BIGINT | Data object logical size. Amount of disk space logically allocated for the table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated for the table (for example, in the case of a logical table truncation). For multi-dimensional clustering (MDC) tables, this size includes the logical size of the block map object. The size returned takes into account full extents that are logically allocated for the table and, for objects created in DMS table spaces, an estimate of the Extent Map Page (EMP) extents. This size represents the logical size of the base table only. Space consumed by LOB data, Long Data, Indexes and XML objects are reported by other columns. |
| DATA_OBJECT_P_SIZE | BIGINT | Data object physical size. Amount of disk space physically allocated for the table, reported in kilobytes. For MDC tables, this size includes the size of the block map object. The size returned takes into account full extents allocated for the table and includes the EMP extents for objects created in DMS table spaces. This size represents the physical size of the base table only. Space consumed by LOB data, Long Data, Indexes and XML objects are reported by other columns. |

Table 231. Information returned by ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO_V97 (continued)

| Column name | Data type | Description |
|---------------------|-----------|---|
| INDEX_OBJECT_L_SIZE | BIGINT | <p>Index object logical size. Amount of disk space logically allocated for the indexes defined on the table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold index data for the table (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for the indexes and, for indexes created in DMS table spaces, an estimate of the EMP extents.</p> <p>For partitioned indexes on partitioned tables, this is the logical size of the index object containing index partitions for the data partition identified by DATA_PARTITION_ID. This value does not take into account nonpartitioned indexes on partitioned tables. For information about both partitioned and nonpartitioned indexes, you can use the ADMIN_GET_INDEX_INFO function.</p> |
| INDEX_OBJECT_P_SIZE | BIGINT | <p>Index object physical size. Amount of disk space physically allocated for the indexes defined on the table, reported in kilobytes. The size returned takes into account full extents allocated for the indexes and includes the EMP extents for indexes created in DMS table spaces.</p> <p>For partitioned indexes on partitioned tables, this is the physical size of the index object containing index partitions for the data partition identified by DATA_PARTITION_ID. This value does not take into account nonpartitioned indexes on partitioned tables. For information about both partitioned and nonpartitioned indexes, you can use the ADMIN_GET_INDEX_INFO function.</p> |
| LONG_OBJECT_L_SIZE | BIGINT | <p>Long object logical size. Amount of disk space logically allocated for long field data in a table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold long field data for the table (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for long field data and, for long field data created in DMS table spaces, an estimate of the EMP extents.</p> |
| LONG_OBJECT_P_SIZE | BIGINT | <p>Long object physical size. Amount of disk space physically allocated for long field data in a table, reported in kilobytes. The size returned takes into account full extents allocated for long field data and includes the EMP extents for long field data created in DMS table spaces.</p> |

Table 231. Information returned by ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO_V97 (continued)

| Column name | Data type | Description |
|----------------------|-------------|---|
| LOB_OBJECT_L_SIZE | BIGINT | LOB object logical size. Amount of disk space logically allocated for LOB data in a table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold LOB data for the table (for example, in the case of a logical table truncation). The size includes space logically allocated for the LOB allocation object. The size returned takes into account full extents that are logically allocated for LOB data and, for LOB data created in DMS table spaces, an estimate of the EMP extents. |
| LOB_OBJECT_P_SIZE | BIGINT | LOB object physical size. Amount of disk space physically allocated for LOB data in a table, reported in kilobytes. The size includes space allocated for the LOB allocation object. The size returned takes into account full extents allocated for LOB data and includes the EMP extents for LOB data created in DMS table spaces. |
| XML_OBJECT_L_SIZE | BIGINT | XML object logical size. Amount of disk space logically allocated for XML data in a table, reported in kilobytes. The logical size is the amount of space that the table knows about. It might be less than the amount of space physically allocated to hold XML data for the table (for example, in the case of a logical table truncation). The size returned takes into account full extents that are logically allocated for XML data and, for XML data created in DMS table spaces, an estimate of the EMP extents. |
| XML_OBJECT_P_SIZE | BIGINT | XML object physical size. Amount of disk space physically allocated for XML data in a table, reported in kilobytes. The size returned takes into account full extents allocated for XML data and includes the EMP extents for XML data created in DMS table spaces. |
| INDEX_TYPE | SMALLINT | Indicates the type of indexes currently in use for the table. Returns: <ul style="list-style-type: none"> • 1 if type-1 indexes are being used. • 2 if type-2 indexes are being used. |
| REORG_PENDING | CHAR(1) | A value of 'Y' indicates that a reorg recommended alter has been applied to the table and a classic (offline) reorg is required. Otherwise 'N' is returned. |
| INPLACE_REORG_STATUS | VARCHAR(10) | Current status of an inplace table reorganization on the table. The status value can be one of the following values: <ul style="list-style-type: none"> • ABORTED (in a PAUSED state, but unable to RESUME; STOP is required) • EXECUTING • NULL (if no inplace reorg has been performed on the table) • PAUSED |

Table 231. Information returned by ADMIN_TABINFO administrative view and the ADMIN_GET_TAB_INFO_V97 (continued)

| Column name | Data type | Description |
|-------------------------|-------------|--|
| LOAD_STATUS | VARCHAR(12) | Current status of a load operation against the table. The status value can be one of the following values: <ul style="list-style-type: none"> • IN_PROGRESS • NULL (if there is no load in progress for the table and the table is not in load pending state) • PENDING |
| READ_ACCESS_ONLY | CHAR(1) | 'Y' if the table is in Read Access Only state, 'N' otherwise. A value of 'N' should not be interpreted as meaning that the table is fully accessible. If a load is in progress or pending, a value of 'Y' means the table data is available for read access, and a value of 'N' means the table is inaccessible. Similarly, if the table status is set integrity pending (refer to SYSCAT.TABLES STATUS column), then a value of 'N' means the table is inaccessible. |
| NO_LOAD_RESTART | CHAR(1) | A value of 'Y' indicates the table is in a partially loaded state that will not allow a load restart. A value of 'N' is returned otherwise. |
| NUM_REORG_REC_ALTERS | SMALLINT | Number of reorg recommend alter operations (for example, alter operations after which a reorganization is required) that have been performed against this table since the last reorganization. |
| INDEXES_REQUIRE_REBUILD | CHAR(1) | For nonpartitioned tables, 'Y' if any of the indexes defined on the table require a rebuild, and 'N' otherwise. For partitioned tables, 'Y' if any index partitions for the data partition identified by DATA_PARTITION_ID require a rebuild, and 'N' otherwise. |
| LARGE_RIDS | CHAR(1) | Indicates whether or not the table is using large row IDs (RIDs) (4 byte page number, 2 byte slot number). A value of 'Y' indicates that the table is using large RIDs and 'N' indicates that it is not using large RIDs. A value of 'P' (pending) will be returned if the table supports large RIDs (that is, the table is in a large table space), but at least one of the indexes for the table has not been reorganized or rebuilt yet, so the table is still using 4 byte RIDs (which means that action must be taken to convert the table or indexes). |
| LARGE_SLOTS | CHAR(1) | Indicates whether or not the table is using large slots (which allows more than 255 rows per page). A value of 'Y' indicates that the table is using large slots and 'N' indicates that it is not using large slots. A value of 'P' (pending) will be returned if the table supports large slots (that is, the table is in a large table space), but there has been no offline table reorganization or table truncation operation performed on the table yet, so it is still using a maximum of 255 rows per page. |
| DICTIONARY_SIZE | BIGINT | Size of the table dictionary, in bytes, used for row compression if a row compression dictionary exists for the table. If a historical dictionary exists, this value is the sum of the current and historical dictionary sizes. |
| BLOCKS_PENDING_CLEANUP | BIGINT | blocks_pending_cleanup - Pending cleanup rolled-out blocks monitor element |

Table 231. Information returned by ADMIN_TABINFO administrative view and the ADMIN_GET_TAB_INFO_V97 (continued)

| Column name | Data type | Description |
|---------------------|-----------|---|
| STATSTYPE | CHAR(1) | <ul style="list-style-type: none"> • 'F' = System fabricated statistics without table or index scan. These statistics are stored in memory and are different from what is stored in the system catalogs. This is a temporary state and eventually full statistics will be gathered by the database system and stored in the system catalogs. • 'A' = System asynchronously gathered statistics. Statistics have been automatically collected by the database system by a background process and stored in the system catalogs. • 'S' = System synchronously gathered statistics. Statistics have been automatically collected by the database system during SQL statement compilation. These statistics are stored in memory and are different from what is stored in the system catalogs. This is a temporary state and eventually the database system stores the statistics in the system catalogs. • 'U' = User gathered statistics. Statistics gathering was initiated by the user through a utility such as RUNSTATS, CREATE INDEX, LOAD, REDISTRIBUTE or by manually updating system catalog statistics. • NULL = unknown type |
| XML_RECORD_TYPE | SMALLINT | <p>Indicates the type of XML record currently in use for the table.</p> <ul style="list-style-type: none"> • 1 if the type-1 (single node) XML record format is being used. • 2 if the type-2 (multi-node) XML record format is being used. • Null if the table has no XML columns. |
| RECLAIMABLE_SPACE | BIGINT | For an MDC table in a DMS table space, this value indicates the amount of disk space that can be reclaimed by running the REORG command with the RECLAIM option. Disk space is reported in kilobytes. For any other table, the value is zero. |
| XML_DICTIONARY_SIZE | BIGINT | Size of the XML dictionary, in bytes, used for data compression if a data compression dictionary exists for the XML storage object. If the table does not contain any XML columns or if a compression dictionary has not been created, the value is 0. |

AM_BASE_RPT_RECOMS - Recommendations for activity reports

The AM_BASE_RPT_RECOMS table function returns recommendations for activity reports used by the activity event monitor.

Syntax

Important: The related activity monitor routines have been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Activity

monitor routines have been deprecated" at http://www.ibm.com/support/knowledgecenter/SSEPGG_10.1.0/com.ibm.db2.luw.wn.doc/doc/i0059660.html.

►►—AM_BASE_RPT_RECOMS—(—report_id—,—client_locale—)————►►

The schema is SYSPROC.

Table function parameters

report_id

An input argument of type INTEGER that specifies a report ID. If the argument is null, recommendations for all available reports are returned.

client_locale

An input argument of type VARCHAR(33) that specifies a client language identifier. If the argument is null or an empty string, the default value is 'En_US' (English). If the message files for the specified locale are not available on the server, 'En_US' is used.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Request recommendations (in English) for the activity event monitor report with an ID of 1. Assume the default client language identifier 'En_US'.

```
SELECT *
  FROM TABLE(SYSPROC.AM_BASE_RPT_RECOMS(1, CAST(NULL AS VARCHAR(33))))
 AS RECOMS
```

Example 2: Request recommendations (in French) for the activity event monitor report with an ID of 12.

```
SELECT *
  FROM TABLE(SYSPROC.AM_BASE_RPT_RECOMS(12, CAST('Fr_FR' AS VARCHAR(33))))
 AS RECOMS
```

Information returned

Table 232. Information returned by the AM_BASE_RPT_RECOMS table function

| Column name | Data type | Description |
|-------------|--------------|--|
| REPORT_ID | INTEGER | The report ID. |
| RECOM_NAME | VARCHAR(256) | The name or short description of the recommendation. |

Table 232. Information returned by the AM_BASE_RPT_RECOMS table function (continued)

| Column name | Data type | Description |
|-------------------|-----------|---|
| RECOM_DESCRIPTION | CLOB(32K) | The detailed description of the recommendation. |

AM_BASE_RPTS - Activity event monitor reports

The AM_BASE_RPTS table function returns activity reports used by the activity event monitor.

Syntax

Important: The related activity monitor routines have been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Activity monitor routines have been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_10.1.0/com.ibm.db2.luw.wn.doc/doc/i0059660.html.

►► AM_BASE_RPTS (—report_id—, —type—, —client_locale—) ◀◀

The schema is SYSPROC.

Table function parameters

report_id

An input argument of type INTEGER that specifies a unique report ID. If the argument is null, reports with any report ID are returned.

type

An input argument of type CHAR(4) that specifies the report type. Valid values are:

APPL Application

STMT SQL statement

TRAN Transaction

CACH Dynamic SQL statement cache

Values can be specified in uppercase or lowercase characters. If the argument is null or an empty string, reports of any type are returned.

client_locale

An input argument of type VARCHAR(33) that specifies a client language identifier. If the argument is null or an empty string, or the message files for the specified locale are not available on the server, 'En_US' is used.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1:

```
SELECT * FROM TABLE(SYSPROC.AM_BASE_RPTS(CAST(NULL AS INTEGER),  
      CAST(NULL AS CHAR(4)), CAST(NULL AS VARCHAR(33)))) AS REPORTS
```

Example 2:

```
SELECT ID, NAME FROM TABLE(SYSPROC.AM_BASE_RPTS(  
      CAST(NULL AS INTEGER), CAST('STMT' AS CHAR(4)), 'En_US'))  
AS REPORTS WHERE TYPE = 'STMT'
```

Information returned

Table 233. Information returned by the AM_BASE_RPTS table function

| Column name | Data type | Description |
|-------------|----------------|--|
| ID | INTEGER | id - cluster caching facility identification monitor element |
| TYPE | CHAR(4) | The report type. Valid values are: APPL, STMT, TRAN, CACH. |
| NAME | VARCHAR(256) | The name or short description of the report. |
| DESCRIPTION | VARCHAR(16384) | The detailed description of the report. |
| SWITCHES | VARCHAR(100) | The monitor switches required for this report. |

AM_DROP_TASK - Delete a monitoring task

The AM_DROP_TASK procedure deletes a monitoring task. It does not return any data.

Syntax

Important: The related activity monitor routines have been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Activity monitor routines have been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_10.1.0/com.ibm.db2.luw.wn.doc/doc/i0059660.html.

►►—AM_DROP_TASK—(—*task_id*—)—————►

The schema is SYSPROC.

Procedure parameter

task_id

An input argument of type INTEGER that specifies a unique monitoring task ID.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

Drop the monitoring task with ID 5.

```
CALL SYSPROC.AM_DROP_TASK(5)
```

AM_GET_LOCK_CHN_TB - Retrieve application lock chain data in a tabular format

The AM_GET_LOCK_CHN_TB procedure returns application lock chain data in tabular format. A lock chain consists of all the applications that the current application is holding up or waiting for, either directly or indirectly.

Syntax

Important: The related activity monitor routines have been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Activity monitor routines have been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_10.1.0/com.ibm.db2.luw.wn.doc/doc/i0059660.html.

►►—AM_GET_LOCK_CHN_TB—(—*agent_id*—)—————►►

The schema is SYSPROC.

Procedure parameters

agent_id

An input argument of type BIGINT that specifies the agent ID of the application for which lock chain data is to be retrieved.

Authorization

- SYSMON authority
- EXECUTE privilege on the AM_GET_LOCK_CHN_TB procedure.

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

Retrieve lock chain information for agent ID 68.

```
CALL SYSPROC.AM_GET_LOCK_CHN_TB(68)
```

Information returned

The procedure returns the following table. Each row of this table represents a lock-wait relationship. The result set also contains a row for each holding-only application; in this case, the HOLDING_AGENT_ID column is null, and the other four columns are for the holding-only application.

Table 234. Information returned by the AM_GET_LOCK_CHN_TB procedure

| Column name | Data Type | Description |
|------------------|--------------|--|
| HOLDING_AGENT_ID | BIGINT | The agent ID of the application holding the lock. |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) monitor element |
| APPL_NAME | VARCHAR(255) | appl_name - Application name monitor element |
| AUTH_ID | VARCHAR(128) | auth_id - Authorization ID monitor element |
| APPL_ID | VARCHAR(64) | appl_id - Application ID monitor element |

AM_GET_LOCK_CHNS - Retrieve lock chain information for a specific application

The AM_GET_LOCK_CHNS procedure returns lock chains for the specified application as a formatted string. A lock chain consists of all the applications that the current application is holding up or waiting for, either directly or indirectly.

Syntax

Important: The related activity monitor routines have been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Activity monitor routines have been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_10.1.0/com.ibm.db2.luw.wn.doc/doc/i0059660.html.

►►—AM_GET_LOCK_CHNS—(—agent_id—,—lock_chains—)————►►

The schema is SYSPROC.

Procedure parameters

agent_id

An input argument of type BIGINT that specifies the agent ID of the application whose lock chains are to be displayed.

lock_chains

An output argument of type CLOB(2M) that shows all the lock chains for the specified application.

Authorization

- SYSMON authority
- EXECUTE privilege on the AM_GET_LOCK_CHNS procedure.

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

```
CALL SYSPROC.AM_GET_LOCK_CHNS(17,?)
Value of output parameters
-----
Parameter Name  : LOCK_CHAINS
Parameter Value : >db2bp.exe (Agent ID: 17) (Auth ID: AMUSERC )

<db2bp.exe (Agent ID: 17) (Auth ID: AMUSERC )
  <db2bp.exe (Agent ID: 18) (Auth ID: AMUSERB )
    <db2bp.exe (Agent ID: 16) (Auth ID: AMUSERA )

Return Status = 0
```

AM_GET_LOCK_RPT - Retrieve application lock details

The AM_GET_LOCK_RPT procedure returns lock details for an application in three output result sets.

Syntax

Important: The related activity monitor routines have been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Activity monitor routines have been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_10.1.0/com.ibm.db2.luw.wn.doc/doc/i0059660.html.

➤➤ AM_GET_LOCK_RPT (—*agent_id*—) ————— ➤➤

The schema is SYSPROC.

Procedure parameter

agent_id

An input argument of type BIGINT that specifies the agent ID of the application whose lock details are to be returned.

Authorization

- SYSMON authority
- EXECUTE privilege on the AM_GET_LOCK_RPT procedure.

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

```
CALL SYSPROC.AM_GET_LOCK_RPT(68)
```

Usage note

The DFT_MON_LOCK monitor switch must be turned on for this procedure to return any information.

Information returned

The procedure returns three result sets: one for application general information; one for locks that the application holds; and one for locks that the application is waiting for.

Table 235. General application information returned by the AM_GET_LOCK_RPT procedure

| Column name | Data Type | Description |
|-----------------|--------------|---|
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| APPL_NAME | VARCHAR(256) | appl_name - Application name |
| PRIMARY_AUTH_ID | VARCHAR(128) | auth_id - Authorization ID |
| APPL_ID | VARCHAR(128) | appl_id - Application ID |
| APPL_STATUS | VARCHAR(22) | appl_status - Application status. This interface returns a text identifier based on the defines in sqlmon.h, and is one of: <ul style="list-style-type: none">• BACKUP• COMMIT_ACT• COMP• CONNECTED• CONNECTPEND• CREATE_DB• DECOUPLED• DISCONNECTPEND• INTR• IOERROR_WAIT• LOAD• LOCKWAIT• QUIESCE_TABLESPACE• RECOMP• REMOTE_RQST• RESTART• RESTORE• ROLLBACK_ACT• ROLLBACK_TO_SAVEPOINT• TEND• THABRT• THCOMT• TPREP• UNLOAD• UOWEXEC• UOWWAIT• WAITFOR_REMOTE |

Table 235. General application information returned by the AM_GET_LOCK_RPT procedure (continued)

| Column name | Data Type | Description |
|-----------------|--------------|---|
| COORD_NODE_NUM | SMALLINT | coord_node - Coordinating node |
| SEQUENCE_NO | VARCHAR(4) | sequence_no - Sequence number |
| CLIENT_PRDID | VARCHAR(128) | client_prdid - Client product/version ID |
| CLIENT_PID | BIGINT | client_pid - Client process ID |
| CLIENT_PLATFORM | VARCHAR(12) | <p>client_platform - Client operating platform. This interface returns a text identifier based on the defines in sqlmon.h,</p> <ul style="list-style-type: none"> • AIX • AIX64 • AS400_DRDA • DOS • DYNIX • HP • HP64 • HPIA • HPIA64 • LINUX • LINUX390 • LINUXIA64 • LINUXPPC • LINUXPPC64 • LINUXX8664 • LINUXZ64 • MAC • MVS_DRDA • NT • NT64 • OS2 • OS390 • SCO • SGI • SNI • SUN • SUN64 • UNKNOWN • UNKNOWN_DRDA • VM_DRDA • VSE_DRDA • WINDOWS • WINDOWS95 |

Table 235. General application information returned by the AM_GET_LOCK_RPT procedure (continued)

| Column name | Data Type | Description |
|----------------------|--------------|---|
| CLIENT_PROTOCOL | VARCHAR(10) | client_protocol - Client communication protocol. This interface returns a text identifier based on the defines in sqlmon.h, <ul style="list-style-type: none"> • CPIC • LOCAL • NPIPE • TCPIP (for Db2) • TCPIP4 • TCPIP6 |
| CLIENT_NNAME | VARCHAR(128) | client_nname - Client name monitor element |
| LOCKS_HELD | BIGINT | locks_held - Locks held |
| LOCK_WAIT_START_TIME | TIMESTAMP | lock_wait_start_time - Lock wait start timestamp |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive lock escalations |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |

Table 236. Locks held information returned by the AM_GET_LOCK_RPT procedure

| Column name | Data Type | Description |
|-------------|--------------|------------------------------------|
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TABNAME | VARCHAR(128) | table_name - Table name |

Table 236. Locks held information returned by the AM_GET_LOCK_RPT procedure (continued)

| Column name | Data Type | Description |
|------------------|-------------|--|
| LOCK_OBJECT_TYPE | VARCHAR(18) | lock_object_type - Lock object type waited on. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • AUTORESIZE_LOCK • AUTOSTORAGE_LOCK • BLOCK_LOCK • EOT_LOCK • INPLACE_REORG_LOCK • INTERNAL_LOCK • INTERNALB_LOCK • INTERNALC_LOCK • INTERNALJ_LOCK • INTERNALL_LOCK • INTERNALO_LOCK • INTERNALQ_LOCK • INTERNALP_LOCK • INTERNALS_LOCK • INTERNALT_LOCK • INTERNALV_LOCK • KEYVALUE_LOCK • ROW_LOCK • SYSBOOT_LOCK • TABLE_LOCK • TABLE_PART_LOCK • TABLESPACE_LOCK • XML_PATH_LOCK • COL_TABLE_SERIALIZE_LOCK |
| LOCK_MODE | VARCHAR(10) | lock_mode - Lock mode. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |

Table 236. Locks held information returned by the AM_GET_LOCK_RPT procedure (continued)

| Column name | Data Type | Description |
|-----------------|-------------|--|
| LOCK_STATUS | VARCHAR(10) | lock_status - Lock status. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • CONV • GRNT |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock escalation |
| LOCK_NAME | VARCHAR(32) | lock_name - Lock name |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

Table 237. Locks wait information returned by the AM_GET_LOCK_RPT procedure

| Column name | Data Type | Description |
|----------------------|--------------|--|
| AGENT_ID_HOLDING_LK | BIGINT | agent_id_holding_lock - Agent ID holding lock |
| APPL_ID_HOLDING_LK | VARCHAR(128) | appl_id_holding_lk - Application ID holding lock |
| LOCK_WAIT_START_TIME | TIMESTAMP | lock_wait_start_time - Lock wait start timestamp |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TABNAME | VARCHAR(128) | table_name - Table name |

Table 237. Locks wait information returned by the AM_GET_LOCK_RPT procedure (continued)

| Column name | Data Type | Description |
|------------------|-------------|--|
| LOCK_OBJECT_TYPE | VARCHAR(18) | lock_object_type - Lock object type waited on. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • AUTORESIZE_LOCK • AUTOSTORAGE_LOCK • BLOCK_LOCK • EOT_LOCK • INPLACE_REORG_LOCK • INTERNAL_LOCK • INTERNALB_LOCK • INTERNALC_LOCK • INTERNALJ_LOCK • INTERNALL_LOCK • INTERNALO_LOCK • INTERNALQ_LOCK • INTERNALP_LOCK • INTERNALS_LOCK • INTERNALT_LOCK • INTERNALV_LOCK • KEYVALUE_LOCK • ROW_LOCK • SYSBOOT_LOCK • TABLE_LOCK • TABLE_PART_LOCK • TABLESPACE_LOCK • XML_PATH_LOCK |
| LOCK_MODE | VARCHAR(10) | lock_mode - Lock mode. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |

Table 237. Locks wait information returned by the AM_GET_LOCK_RPT procedure (continued)

| Column name | Data Type | Description |
|---------------------|-------------|---|
| LOCK_MODE_REQUESTED | VARCHAR(10) | lock_mode_requested - Lock mode requested. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock escalation |

AM_GET_RPT - Retrieve activity monitor data

The AM_GET_RPT procedure returns activity monitor data for a report.

Syntax

Important: The related activity monitor routines have been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Activity monitor routines have been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_10.1.0/com.ibm.db2.luw.wn.doc/doc/i0059660.html.

►►—AM_GET_RPT—(—partition—,—report_id—,—appl_filter—,—max_number—)—►►

The schema is SYSPROC.

Procedure parameters

partition

An input argument of type INTEGER that specifies a database partition number. Valid values are -2 (denoting all database partitions) and the database partition number of any existing database partition.

report_id

An input argument of type INTEGER that specifies a unique report ID.

appl_filter

An input argument of type CLOB(32K) that specifies an application filter. An application filter is a search condition involving any or all of the three columns AGENT_ID, APPL_NAME, and PRIMARY_AUTH_ID, where AGENT_ID and PRIMARY_AUTH_ID are integers, and APPL_NAME is a character string. If the argument is null or an empty string, no filtering is performed.

max_number

An input argument of type INTEGER that specifies the maximum number of

applications, statements, or transactions that are to be displayed. If the argument is null, all applications, statements, and transactions will be displayed.

Authorization

- SYSMON authority
- EXECUTE privilege on the AM_GET_RPT procedure.

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

```
CALL SYSPROC.AM_GET_RPT(-2, 18,
  CAST('AGENT_ID=29 AND PRIMARY_AUTH_ID <> ''dbuser'' AND APPL_NAME LIKE ''db2%'''
    AS CLOB(32K)), 100)
```

Usage note

The result set returned is different for each report id. This procedure is intended to support the discontinued Activity Monitor graphical tool. To build reports that can be parsed, snapshot administrative SQL routines and views should be used instead. To use this procedure, the DFT_MON_LOCK monitor switch must be turned on.

AM_SAVE_TASK - Create or modify a monitoring task

The AM_SAVE_TASK procedure creates or modifies a monitoring task.

Syntax

Important: The related activity monitor routines have been deprecated in Version 10.1 and might be removed in a future release. For more information, see “Activity monitor routines have been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_10.1.0/com.ibm.db2.luw.wn.doc/doc/i0059660.html.

```
►►AM_SAVE_TASK—(—mode—,—task_id—,—task_name—,—appl_filter—,——————►
►—show_lock_chains—,—report_ids—)—————►◄
```

The schema is SYSPROC.

Procedure parameters

mode

An input argument of type CHAR(1) that specifies whether to create a new monitoring task ('C') or to modify an existing monitoring task ('M').

task_id

An input argument of type INTEGER that specifies a unique monitoring task ID. When *mode* is 'C', any specified input for *task_id* is ignored. An ID for the new monitoring task will be generated by the procedure and returned in the output. When *mode* is 'M', specifies the ID of the monitoring task that is being modified.

task_name

An input argument of type VARCHAR(128) that specifies a name or short description for a monitoring task.

appl_filter

An input argument of type CLOB(32K) that specifies an application filter. An application filter is a search condition involving any or all of the three columns AGENT_ID, APPL_NAME, and AUTH_ID, where AGENT_ID and AUTH_ID are integers, and APPL_NAME is a character string. If the argument is null or an empty string, no filtering is performed.

show_lock_chains

An input argument of type CHAR(1) that specifies whether lock chains are to be shown. Valid values are 'Y' and 'N'. If the argument is null, lock chains are not to be shown.

report_ids

An input argument of type VARCHAR(3893) that specifies one or more report IDs separated by commas.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

Example:

```
CALL SYSPROC.AM_SAVE_TASK('M',11,'Task ABC',CAST (NULL AS CLOB(32K)),  
    'N','1,2,4,8,9,12')
```

APPL_PERFORMANCE administrative view - Retrieve percentage of rows selected for an application

The APPL_PERFORMANCE administrative view displays information about the percentage of rows selected by an application.

The information returned is for all database partitions for the currently connected database. This view can be used to look for applications that might be performing large table scans or to look for potentially troublesome queries.

Important: The APPL_PERFORMANCE administrative view is deprecated and has been replaced by the “MON_CONNECTION_SUMMARY - Retrieve metrics for all connections” on page 430.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the APPL_PERFORMANCE administrative view
- CONTROL privilege on the APPL_PERFORMANCE administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve the report on application performance.

```
SELECT SNAPSHOT_TIMESTAMP, SUBSTR(AUTHID,1,10) AS AUTHID,  
       SUBSTR(APPL_NAME,1,10) AS APPL_NAME,AGENT_ID,  
       PERCENT_ROWS_SELECTED, DBPARTITIONNUM  
FROM SYSIBMADM.APPL_PERFORMANCE
```

The following is an example of output for this query.

```
SNAPSHOT_TIMESTAMP      AUTHID      APPL_NAME ...  
-----  
2006-01-07-17.01.15.966668 JESSICAE  db2bp.exe ...  
2006-01-07-17.01.15.980278 JESSICAE  db2taskd ...  
2006-01-07-17.01.15.980278 JESSICAE  db2bp.exe ...  
...  
3 record(s) selected.    ...
```

Output for this query (continued).

```
... AGENT_ID      PERCENT_ROWS_SELECTED DBPARTITIONNUM  
... -----  
...          67              -              1  
...          68              -              0  
...          67             57.14             0  
...
```

Information returned

Table 238. Information returned by the APPL_PERFORMANCE administrative view

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| AUTHID | VARCHAR(128) | auth_id - Authorization ID |

Table 238. Information returned by the APPL_PERFORMANCE administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|--------------|--|
| APPL_NAME | VARCHAR(256) | appl_name - Application name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| PERCENT_ROWS_SELECTED | DECIMAL(5,2) | The percent of rows read from disk that were actually returned to the application.
Note: The percentage shown will not be greater than 100.00 percent. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

APPLICATIONS administrative view - Retrieve connected database application information

The APPLICATIONS administrative view returns information about the connected database applications.

The view is an SQL interface for the **LIST APPLICATIONS SHOW DETAIL CLP** command, but only for the currently connected database. Its information is based on the SNAPAPPL_INFO administrative view.

Important: The SNAPAPPL administrative view and the SNAP_GET_APPL table function are deprecated and have been replaced by the “MON_GET_CONNECTION table function - Get connection metrics” on page 588.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the APPLICATIONS administrative view
- CONTROL privilege on the APPLICATIONS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Example 1: List information for all the active applications in the single-partitioned database SAMPLE.

```
SELECT AGENT_ID, SUBSTR(APPL_NAME,1,10) AS APPL_NAME, AUTHID,  
       APPL_STATUS FROM SYSIBMADM.APPLICATIONS WHERE DB_NAME = 'SAMPLE'
```

The following is an example of output for this query.

| AGENT_ID | APPL_NAME | AUTHID | APPL_STATUS |
|----------|-----------|----------|-------------|
| 23 | db2bp.exe | JESSICAE | UOWEXEC |

1 record(s) selected.

Example 2: List the number of agents per application on database partition 0 for the multi-partition database SAMPLE.

```
SELECT SUBSTR(APPL_NAME, 1, 10) AS APPL_NAME, COUNT(*) AS NUM  
FROM SYSIBMADM.APPLICATIONS WHERE DBPARTITIONNUM = 0  
AND DB_NAME = 'SAMPLE' GROUP BY APPL_NAME
```

The following is an example of output for this query.

| APPL_NAME | NUM |
|-----------|-----|
| db2bp.exe | 3 |
| javaw.exe | 1 |

2 record(s) selected.

Usage notes

The view does not support the **GLOBAL** syntax available from the CLP. However, aggregation can be done using SQL aggregation functions as data from all database partitions is returned from the view.

Information returned

Table 239. Information returned by the APPLICATIONS administrative view

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| CLIENT_DB_ALIAS | VARCHAR(128) | client_db_alias - Database alias used by application |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| APPL_NAME | VARCHAR(256) | appl_name - Application name |
| AUTHID | VARCHAR(128) | auth_id - Authorization ID |
| APPL_ID | VARCHAR(128) | appl_id - Application ID |

Table 239. Information returned by the APPLICATIONS administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| APPL_STATUS | VARCHAR(22) | <p>appl_status - Application status. This interface returns a text identifier based on defines in sqlmon.h, and is one of:</p> <ul style="list-style-type: none"> • BACKUP • COMMIT_ACT • COMP • CONNECTED • CONNECTPEND • CREATE_DB • DECOUPLED • DISCONNECTPEND • INTR • IOERROR_WAIT • LOAD • LOCKWAIT • QUIESCE_TABLESPACE • RECOMP • REMOTE_RQST • RESTART • RESTORE • ROLLBACK_ACT • ROLLBACK_TO_SAVEPOINT • TEND • THABRT • THCOMT • TPREP • UNLOAD • UOWEXEC • UOWWAIT • WAITFOR_REMOTE |
| STATUS_CHANGE_TIME | TIMESTAMP | status_change_time - Application status change time |
| SEQUENCE_NO | VARCHAR(4) | sequence_no - Sequence number |
| CLIENT_PRDID | VARCHAR(128) | client_prdid - Client product/version ID |
| CLIENT_PID | BIGINT | client_pid - Client process ID |

Table 239. Information returned by the APPLICATIONS administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------|-------------|--|
| CLIENT_PLATFORM | VARCHAR(12) | <p>client_platform - Client operating platform. This interface returns a text identifier based on defines in sqlmon.h, and is one of:</p> <ul style="list-style-type: none"> • AIX • AIX64 • AS400_DRDA • DOS • DYNIX • HP • HP64 • HPIA • HPIA64 • LINUX • LINUX390 • LINUXIA64 • LINUXPPC • LINUXPPC64 • LINUXX8664 • LINUXZ64 • MAC • MVS_DRDA • NT • NT64 • OS2 • OS390 • SCO • SGI • SNI • SUN • SUN64 • UNKNOWN • UNKNOWN_DRDA • VM_DRDA • VSE_DRDA • WINDOWS • WINDOWS95 |

Table 239. Information returned by the APPLICATIONS administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|--------------|---|
| CLIENT_PROTOCOL | VARCHAR(10) | client_protocol - Client communication protocol. This interface returns a text identifier based on the defines in sqlmon.h, <ul style="list-style-type: none"> • CPIC • LOCAL • NPIPE • TCPIP • TCPIP4 • TCPIP6 |
| CLIENT_NNAME | VARCHAR(128) | client_nname - Client name monitor element |
| COORD_NODE_NUM | SMALLINT | coord_node - Coordinating node |
| COORD_AGENT_PID | BIGINT | coord_agent_pid - Coordinator agent |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of associated agents |
| TPMON_CLIENT_USERID | VARCHAR(256) | tpmon_client_userid - TP monitor client user ID |
| TPMON_CLIENT_WKSTN | VARCHAR(256) | tpmon_client_wkstn - TP monitor client workstation name |
| TPMON_CLIENT_APP | VARCHAR(256) | tpmon_client_app - TP monitor client application name |
| TPMON_ACC_STR | VARCHAR(255) | tpmon_acc_str - TP monitor client accounting string |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| COORD_MEMBER | SMALLINT | coord_member - Coordinator member monitor element |
| COORD_DBPARTITIONNUM | SMALLINT | The coordinating database partition number. |

APPLICATION_ID

The APPLICATION_ID function returns the application ID of the current connection.

The data type of the result is VARCHAR(128).

The value returned by the function is unique within a 100-year interval and valid only for the duration of the connection established before calling the function.

Note: This scalar function has been deprecated and replaced by the MON_GET_APPLICATION_ID scalar function.

Syntax

►►APPLICATION_ID—(—)◄◄

The schema is SYSFUN.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT APPLICATION_ID() AS APPL_ID FROM SYSIBM.SYSDUMMY1
```

BP_HITRATIO administrative view - Retrieve bufferpool hit ratio information

The BP_HITRATIO administrative view returns bufferpool hit ratios, including total hit ratio, data hit ratio, XDA hit ratio and index hit ratio, for all bufferpools and all database partitions in the currently connected database.

Important: The BP_HITRATIO administrative view is deprecated and has been replaced by the “MON_BP_UTILIZATION administrative view - Retrieve metrics for bufferpools” on page 422.

Note: This administrative view works only in Db2 environments without the IBM Db2 pureScale Feature. For information about calculating hit ratios in a Db2 pureScale environment, see “Calculating buffer pool hit ratios in a Db2 pureScale environment”, in the *Database Monitoring Guide and Reference*.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the BP_HITRATIO administrative view
- CONTROL privilege on the BP_HITRATIO administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON

- SYSCTRL
- SYSMaint
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a report for all bufferpools in the connected database.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, SUBSTR(BP_NAME,1,14) AS BP_NAME,
       TOTAL_HIT_RATIO_PERCENT, DATA_HIT_RATIO_PERCENT,
       INDEX_HIT_RATIO_PERCENT, XDA_HIT_RATIO_PERCENT, DBPARTITIONNUM
FROM SYSIBMADM.BP_HITRATIO ORDER BY DBPARTITIONNUM
```

The following is an example of output for this query.

| DB_NAME | BP_NAME | TOTAL_HIT_RATIO_PERCENT | DATA_HIT_RATIO_PERCENT | ... |
|---------|----------------|-------------------------|------------------------|-----|
| TEST | IBMDEFAULTBP | 63.09 | 68.94 | ... |
| TEST | IBMSYSTEMBP4K | - | - | ... |
| TEST | IBMSYSTEMBP8K | - | - | ... |
| TEST | IBMSYSTEMBP16K | - | - | ... |
| TEST | IBMSYSTEMBP32K | - | - | ... |

Output for this query (continued).

| ... | INDEX_HIT_RATIO_PERCENT | XDA_HIT_RATIO_PERCENT | DBPARTITIONNUM |
|-----|-------------------------|-----------------------|----------------|
| ... | 43.20 | - | 0 |
| ... | - | - | 0 |
| ... | - | - | 0 |
| ... | - | - | 0 |
| ... | - | - | 0 |

Usage notes

The ratio of physical reads to total reads gives the hit ratio for the bufferpool. The lower the hit ratio, the more the data is being read from disk rather than the cached buffer pool which can be a more costly operation.

Information returned

Table 240. Information returned by the BP_HITRATIO administrative view

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | Timestamp when the report was requested. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| BP_NAME | VARCHAR(128) | bp_name - Buffer pool name |
| TOTAL_LOGICAL_READS | BIGINT | Total logical reads (index, XDA and data) in the bufferpool. |
| TOTAL_PHYSICAL_READS | BIGINT | Total physical reads (index, XDA and data) in the bufferpool. |
| TOTAL_HIT_RATIO_PERCENT | DECIMAL(5,2) | Total hit ratio (index, XDA and data reads). |

Table 240. Information returned by the BP_HITRATIO administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|--|
| DATA_LOGICAL_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| DATA_PHYSICAL_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| DATA_HIT_RATIO_PERCENT | DECIMAL(5,2) | Data hit ratio. |
| INDEX_LOGICAL_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| INDEX_PHYSICAL_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| INDEX_HIT_RATIO_PERCENT | DECIMAL(5,2) | Index hit ratio. |
| XDA_LOGICAL_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| XDA_PHYSICAL_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| XDA_HIT_RATIO_PERCENT | DECIMAL(5,2) | Auxiliary storage objects hit ratio. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

BP_READ_IO administrative view - Retrieve bufferpool read performance information

The BP_READ_IO administrative view returns bufferpool read performance information. This view can be used to look at each bufferpool to see how effective the prefetchers are.

Important: The BP_READ_IO administrative view is deprecated and has been replaced by the “MON_BP_UTILIZATION administrative view - Retrieve metrics for bufferpools” on page 422.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the BP_READ_IO administrative view
- CONTROL privilege on the BP_READ_IO administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT

- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve total physical reads and average read time for all bufferpools on all partitions of the currently connected database.

```
SELECT SUBSTR(BP_NAME, 1, 15) AS BP_NAME, TOTAL_PHYSICAL_READS,
       AVERAGE_READ_TIME_MS, DBPARTITIONNUM
FROM SYSIBMADM.BP_READ_IO ORDER BY DBPARTITIONNUM
```

The following is an example of output for this query.

| BP_NAME | TOTAL_PHYSICAL_READS | AVERAGE_READ_TIME_MS | DBPARTITIONNUM |
|----------------|----------------------|----------------------|----------------|
| IBMDEFAULTBP | 811 | 4 | 0 |
| IBMSYSTEMBP4K | 0 | - | 0 |
| IBMSYSTEMBP8K | 0 | - | 0 |
| IBMSYSTEMBP16K | 0 | - | 0 |
| IBMDEFAULTBP | 34 | 0 | 1 |
| IBMSYSTEMBP4K | 0 | - | 1 |
| IBMSYSTEMBP8K | 0 | - | 1 |
| IBMDEFAULTBP | 34 | 0 | 2 |
| IBMSYSTEMBP4K | 0 | - | 2 |
| IBMSYSTEMBP8K | 0 | - | 2 |

10 record(s) selected.

Information returned

Table 241. Information returned by the BP_READ_IO administrative view

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | Date and time the report was generated. |
| BP_NAME | VARCHAR(128) | bp_name - Buffer pool name |
| TOTAL_PHYSICAL_READS | BIGINT | Total physical reads. |
| AVERAGE_READ_TIME_MS | BIGINT | Average read time in milliseconds. |
| TOTAL_ASYNC_READS | BIGINT | Total asynchronous reads. |
| AVERAGE_ASYNC_READ_TIME_MS | BIGINT | Average asynchronous read time in milliseconds. |
| TOTAL_SYNC_READS | BIGINT | Total synchronous reads. |
| AVERAGE_SYNC_READ_TIME_MS | BIGINT | Average synchronous read time in milliseconds. |

Table 241. Information returned by the BP_READ_IO administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|--------------|--|
| PERCENT_SYNC_READS | DECIMAL(5,2) | Percentage of pages read synchronously without prefetching. If many of the applications are reading data synchronously without prefetching then the system might not be tuned optimally. |
| ASYNC_NOT_READ_PERCENT | DECIMAL(5,2) | Percentage of pages read asynchronously from disk, but never accessed by a query. If too many pages are read asynchronously from disk into the bufferpool, but no query ever accesses those pages, then the prefetching might degrade performance. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

BP_WRITE_IO administrative view - Retrieve bufferpool write performance information

The BP_WRITE_IO administrative view returns bufferpool write performance information per bufferpool.

Important: The BP_WRITE_IO administrative view is deprecated and has been replaced by the “MON_BP_UTILIZATION administrative view - Retrieve metrics for bufferpools” on page 422.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the BP_WRITE_IO administrative view
- CONTROL privilege on the BP_WRITE_IO administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve total writes and average write time for all bufferpools on all database partitions of the currently connected database.

```
SELECT SUBSTR(BP_NAME, 1, 15) AS BP_NAME, TOTAL_WRITES,
       AVERAGE_WRITE_TIME_MS, DBPARTITIONNUM
FROM SYSIBMADM.BP_WRITE_IO ORDER BY DBPARTITIONNUM
```

The following is an example of output for this query.

| BP_NAME | TOTAL_WRITES | AVERAGE_WRITE_TIME_MS | DBPARTITIONNUM |
|----------------|--------------|-----------------------|----------------|
| IBMDEFAULTBP | 11 | 5 | 0 |
| IBMSYSTEMBP4K | 0 | - | 0 |
| IBMSYSTEMBP8K | 0 | - | 0 |
| IBMSYSTEMBP16K | 0 | - | 0 |
| IBMSYSTEMBP32K | 0 | - | 0 |
| IBMDEFAULTBP | 0 | - | 1 |
| IBMSYSTEMBP4K | 0 | - | 1 |
| IBMSYSTEMBP8K | 0 | - | 1 |
| IBMDEFAULTBP | 0 | - | 2 |
| IBMSYSTEMBP4K | 0 | - | 2 |
| IBMSYSTEMBP8K | 0 | - | 2 |

11 record(s) selected.

Information returned

Table 242. Information returned by the BP_WRITE_IO administrative view

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time the report was generated. |
| BP_NAME | VARCHAR(128) | bp_name - Buffer pool name |
| TOTAL_WRITES | BIGINT | Total writes. |
| AVERAGE_WRITE_TIME_MS | BIGINT | Average write time in milliseconds. |
| TOTAL_ASYNC_WRITES | BIGINT | Total asynchronous writes. |
| PERCENT_WRITES_ASYNC | BIGINT | Percent of writes that are asynchronous. |

Table 242. Information returned by the BP_WRITE_IO administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------------|-----------|--|
| AVERAGE_ASYNC_WRITE_TIME_MS | BIGINT | Average asynchronous write time in milliseconds. |
| TOTAL_SYNC_WRITES | BIGINT | Total synchronous writes. |
| AVERAGE_SYNC_WRITE_TIME_MS | BIGINT | Average synchronous write time in milliseconds. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

CONTAINER_UTILIZATION administrative view - Retrieve table space container and utilization information

The CONTAINER_UTILIZATION administrative view returns information about table space containers and utilization rates.

It retrieve a similar report to the LIST TABLESPACES command on a single partitioned database. Its information is based on the SNAPCONTAINER administrative view.

Important: The CONTAINER_UTILIZATION administrative view is deprecated and has been replaced by the “MON_GET_CONTAINER table function - Get table space container metrics” on page 616.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the CONTAINER_UTILIZATION administrative view
- CONTROL privilege on the CONTAINER_UTILIZATION administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of all table spaces containers in the connected single partition database, including information about the total and usable pages as well as their accessibility status.

```
SELECT SUBSTR(TBSP_NAME,1,20) AS TBSP_NAME, INT(TBSP_ID) AS TBSP_ID,
       SUBSTR(CONTAINER_NAME,1,45) AS CONTAINER_NAME, INT(CONTAINER_ID)
       AS CONTAINER_ID, CONTAINER_TYPE, INT(TOTAL_PAGES) AS TOTAL_PAGES,
       INT(USABLE_PAGES) AS USABLE_PAGES, ACCESSIBLE
FROM SYSIBMADM.CONTAINER_UTILIZATION
```

The following is an example of output for this query.

| TBSP_NAME | TBSP_ID | CONTAINER_NAME | ... |
|------------------|---------|---|-----|
| SYSCATSPACE | 0 | D:\DB2\NODE0000\SQL00001\SQLT0000.0 | ... |
| TEMPSPACE1 | 1 | D:\DB2\NODE0000\SQL00001\SQLT0001.0 | ... |
| USERSPACE1 | 2 | D:\DB2\NODE0000\SQL00001\SQLT0002.0 | ... |
| SYSTOOLSPACE | 3 | D:\DB2\NODE0000\SQL00001\SYSTOOLSPACE | ... |
| SYSTOOLSTMPSPACE | 4 | D:\DB2\NODE0000\SQL00001\SYSTOOLSTMPSPACE | ... |

5 record(s) selected.

Output for this query (continued).

| ... | CONTAINER_ID | CONTAINER_TYPE | TOTAL_PAGES | USABLE_PAGES | ACCESSIBLE |
|-----|--------------|----------------|-------------|--------------|------------|
| ... | 0 | PATH | 0 | 0 | 1 |
| ... | 0 | PATH | 0 | 0 | 1 |
| ... | 0 | PATH | 0 | 0 | 1 |
| ... | 0 | PATH | 0 | 0 | 1 |
| ... | 0 | PATH | 0 | 0 | 1 |

Information returned

The BUFFERPOOL snapshot monitor switch must be enabled at the database manager configuration for the file system information to be returned.

Table 243. Information returned by the CONTAINER_UTILIZATION administrative view

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container name |
| CONTAINER_ID | BIGINT | container_id - Container identification |

Table 243. Information returned by the *CONTAINER_UTILIZATION* administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------|-------------|--|
| CONTAINER_TYPE | VARCHAR(16) | container_type - Container type

This is a text identifier based on the defines in sqlutil.h and is one of: <ul style="list-style-type: none"> • DISK_EXTENT_TAG • DISK_PAGE_TAG • FILE_EXTENT_TAG • FILE_PAGE_TAG • PATH |
| TOTAL_PAGES | BIGINT | container_total_pages - Total pages in container |
| USABLE_PAGES | BIGINT | container_usable_pages - Usable pages in container |
| ACCESSIBLE | SMALLINT | container_accessible - Accessibility of container |
| STRIPE_SET | BIGINT | container_stripe_set - Stripe set |
| FS_ID | VARCHAR(22) | fs_id - Unique file system identification number |
| FS_TOTAL_SIZE_KB | BIGINT | fs_total_size - Total size of a file system. This interface returns the value in KB. |
| FS_USED_SIZE_KB | BIGINT | fs_used_size - Amount of space used on a file system. This interface returns the value in KB. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

DB_PARTITIONS

The DB_PARTITIONS table function returns the contents of the db2nodes.cfg file in table format.

Note: This table function has been deprecated and replaced by the DB2_MEMBER and DB2_CF administrative views and DB2_GET_INSTANCE_INFO table function.

Syntax

►►DB_PARTITIONS—(—)—————►

The schema is SYSPROC.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority

- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Table function parameters

The function has no input parameters.

Examples

Retrieve information from a 4 member partitioned database instance.

```
SELECT * FROM TABLE(DB_PARTITIONS()) as T
```

The following is an example of the output from this query:

| PARTITION_NUMBER | HOST_NAME | PORT_NUMBER | SWITCHNAME |
|------------------|-----------|-------------|------------|
| 0 | so1 | 0 | so1-ib0 |
| 1 | so2 | 0 | so2-ib0 |
| 2 | so3 | 0 | so3-ib0 |
| 3 | so4 | 0 | so4-ib0 |

4 record(s) selected.

In a Db2 pureScale environment, retrieve information from a 3 member and 1 cluster caching facility Db2 pureScale instance.

```
SELECT * FROM TABLE(DB_PARTITIONS()) as T
```

The following is an example of the output from this query:

| PARTITION_NUMBER | HOST_NAME | PORT_NUMBER | SWITCHNAME |
|------------------|-----------|-------------|------------|
| 0 | so1 | 0 | so1-ib0 |
| 0 | so2 | 0 | so2-ib0 |
| 0 | so3 | 0 | so3-ib0 |

3 record(s) selected.

Usage notes

For Db2 Enterprise Server Edition and in a partitioned database environment, the DB_PARTITIONS table function returns one row for each entry in the db2nodes.cfg file.

In a Db2 pureScale environment, the DB_PARTITIONS table function returns multiple rows, with the following information in the columns:

- The PARTITION_NUMBER column always contains 0.
- The remaining columns show information for the entry in the db2nodes.cfg file for the current member.

Information returned

Table 244. Information returned by the DB_PARTITIONS table function

| Column name | Data type | Description |
|------------------|--------------|--|
| PARTITION_NUMBER | SMALLINT | partition_number - Partition Number monitor element |
| HOST_NAME | VARCHAR(256) | host_name - Host name monitor element |
| PORT_NUMBER | SMALLINT | The port number for the database partition server. |
| SWITCH_NAME | VARCHAR(128) | The name of a high speed interconnect, or switch, for database partition communications. |

ENV_SYS_RESOURCES administrative view - Return system information

The ENV_SYS_RESOURCES administrative view returns operating system, CPU, memory and other information related to the system.

Note: The ENV_SYS_RESOURCES administrative view and the associated ENV_GET_SYS_RESOURCES table function are deprecated and have been replaced by the ENV_GET_SYSTEM_RESOURCES table function.

The ENV_SYS_RESOURCES administrative view returns operating system, CPU, memory, and other information related to the system.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the ENV_SYS_RESOURCES administrative view
- CONTROL privilege on the ENV_SYS_RESOURCES administrative view
- DATAACCESS authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

```
SELECT SUBSTR(NAME,1,20) AS NAME, SUBSTR(VALUE,1,10) AS VALUE,  
       SUBSTR(DATATYPE,1,10) AS DATATYPE, DBPARTITIONNUM  
FROM SYSIBMADM.ENV_SYS_RESOURCES  
WHERE SUBSTR(NAME,1,8)='CPU_LOAD' OR NAME='CPU_USAGE_TOTAL'
```

The following is an example of output from this query.

| NAME | VALUE | DATATYPE | DBPARTITIONNUM |
|-----------------|----------|----------|----------------|
| CPU_LOAD_SHORT | 0.044052 | DECIMAL | 0 |
| CPU_LOAD_MEDIUM | 0.087250 | DECIMAL | 0 |

| | | | |
|-----------------|----------|----------|---|
| CPU_LOAD_LONG | 0.142059 | DECIMAL | 0 |
| CPU_USAGE_TOTAL | 7 | SMALLINT | 0 |

4 record(s) selected.

Usage Notes

To return accurate results, the ENV_SYS_RESOURCES administrative view takes more than 1 second on AIX, HP, and Windows operating systems.

ENV_SYS_RESOURCES administrative view metadata

Table 245. ENV_SYS_RESOURCES administrative view metadata

| Column name | Data type | Description |
|----------------|---------------|--|
| NAME | VARCHAR(128) | Name of the attribute. See Table 246 for possible values.
Note: Some attributes might not be available depending on the operating system and hardware configuration at the server. |
| VALUE | VARCHAR(1024) | The value of the attribute. |
| DATATYPE | VARCHAR(128) | Attribute data type. |
| UNIT | VARCHAR(128) | Unit used for the VALUE column if applicable. NULL is returned if not applicable. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

Table 246. Possible values for the NAME column

| Information type | Name | Data types | Description | Platforms that return this information | UNIT |
|------------------|------------------------|--------------|---|--|------|
| Operating system | OS_NAME | VARCHAR(256) | Name of the operating system software. | All | NULL |
| | HOST_NAME | VARCHAR(256) | Host name of the system. | All | NULL |
| | OS_VERSION | VARCHAR(256) | Version of the operating system.
For example, AIX: 4.3 version = 4. | All | NULL |
| | OS_RELEASE | VARCHAR(256) | Release of the operating system.
For example, AIX: 4.3 release = 3. | All | NULL |
| | MACHINE_IDENTIFICATION | VARCHAR(256) | Machine hardware identification. | All | NULL |
| | OS_LEVEL | VARCHAR(256) | Maintenance level of the current version and release. For example, LINUX: 2.4.9, level = 9. | Linux | NULL |

Table 246. Possible values for the NAME column (continued)

| Information type | Name | Data types | Description | Platforms that return this information | UNIT |
|------------------|----------------------|------------|--|--|------|
| CPU | CPU_TOTAL | BIGINT | Total number of CPUs. | All | NULL |
| | CPU_ONLINE | BIGINT | Number of CPUs online. | All | NULL |
| | CPU_CONFIGURED | BIGINT | Number of CPUs configured. | All | NULL |
| | CPU_SPEED | BIGINT | Speed of CPUs. | All | MHz |
| | CPU_TIMEBASE | BIGINT | Frequency of timebase register increment. | Linux
PowerPC® | Hz |
| | CPU_HMT_DEGREE | BIGINT | On systems that support hardware multithreading (HMT), this is the number of processors that a physical processor will appear to the operating system as. On non-HMT systems, this value is 1. On HMT systems, "total" will reflect the number of logical CPUs. To get the number of physical CPUs, divide the "total" by "threadingDegree". | All | NULL |
| | CPU_CORES_PER_SOCKET | BIGINT | Number of CPU cores per socket. On single core systems this value is 1. | All | NULL |
| Physical memory | MEMORY_TOTAL | BIGINT | Total size of physical memory. | All | MB |
| | MEMORY_FREE | BIGINT | Amount of free physical memory. | All | MB |
| | MEMORY_SWAP_TOTAL | BIGINT | Total amount of swap space. | All | MB |
| | MEMORY_SWAP_FREE | BIGINT | Amount of free swap space. | All | MB |
| Virtual memory | VIRTUAL_MEM_TOTAL | BIGINT | Total amount of virtual memory on the system. | All | MB |
| | VIRTUAL_MEM_RESERVED | BIGINT | Amount of reserved virtual memory. | All | MB |
| | VIRTUAL_MEM_FREE | BIGINT | Amount of virtual memory free. | All | MB |

Table 246. Possible values for the NAME column (continued)

| Information type | Name | Data types | Description | Platforms that return this information | UNIT |
|------------------|-----------------|------------|--|--|---------|
| CPU load | CPU_LOAD_SHORT | DECIMAL | Shortest period duration. For example, load samples over last 5 minutes. | All except Windows operating systems | NULL |
| | CPU_LOAD_MEDIUM | DECIMAL | Medium period duration. For example, load samples over last 10 minutes. | All except Windows operating systems | NULL |
| | CPU_LOAD_LONG | DECIMAL | Long period duration. For example, load samples over last 15 minutes. | All except Windows operating systems | NULL |
| | CPU_USAGE_TOTAL | SMALLINT | Percentage of overall CPU usage of the machine. | All | Percent |

ENV_GET_SYS_RESOURCES table function - return system information

The ENV_GET_SYS_RESOURCES table function returns operating system, CPU, memory, and other information related to the system.

Important: The ENV_GET_SYS_RESOURCES table function has been deprecated and replaced by the “ENV_GET_SYSTEM_RESOURCES table function - Return system information” on page 395.

Syntax

►►—ENV_GET_SYS_RESOURCES—(—)—————►►

The schema is SYSPROC.

Authorization

EXECUTE privilege on the ENV_GET_SYS_RESOURCES table function.

Examples

The following example describes a sample query that uses the ENV_GET_SYS_RESOURCES table function, and the resulting output.

```
SELECT SUBSTR(NAME,1,20) AS NAME, SUBSTR(VALUE,1,10) AS VALUE,
SUBSTR(DATATYPE,1,10) AS DATATYPE, DBPARTITIONNUM
FROM TABLE(SYSPROC.ENV_GET_SYS_RESOURCES()) AS T
```

| NAME | VALUE | DATATYPE | DBPARTITIONNUM |
|---------|-------|-------------|----------------|
| OS_NAME | Linux | VARCHAR(25) | 0 |

| | | | |
|----------------------|----------|-------------|---|
| HOST_NAME | host1 | VARCHAR(25) | 0 |
| OS_VERSION | 2 | VARCHAR(25) | 0 |
| OS_RELEASE | 6 | VARCHAR(25) | 0 |
| MACHINE_IDENTIFICATI | x86_64 | VARCHAR(25) | 0 |
| OS_LEVEL | 5 | VARCHAR(25) | 0 |
| CPU_TOTAL | 4 | BIGINT | 0 |
| CPU_ONLINE | 4 | BIGINT | 0 |
| CPU_CONFIGURED | 4 | BIGINT | 0 |
| CPU_SPEED | 2599 | BIGINT | 0 |
| CPU_HMT_DEGREE | 1 | BIGINT | 0 |
| CPU_CORES_PER_SOCKET | 1 | BIGINT | 0 |
| MEMORY_TOTAL | 7778 | BIGINT | 0 |
| MEMORY_FREE | 2474 | BIGINT | 0 |
| MEMORY_SWAP_TOTAL | 2056 | BIGINT | 0 |
| MEMORY_SWAP_FREE | 1964 | BIGINT | 0 |
| VIRTUAL_MEM_TOTAL | 9834 | BIGINT | 0 |
| VIRTUAL_MEM_FREE | 4438 | BIGINT | 0 |
| CPU_LOAD_SHORT | 0.020000 | DECIMAL | 0 |
| CPU_LOAD_MEDIUM | 0.030000 | DECIMAL | 0 |
| CPU_LOAD_LONG | 0.040000 | DECIMAL | 0 |
| CPU_USAGE_TOTAL | 0 | SMALLINT | 0 |

22 record(s) selected.

Usage Notes

To return accurate results, the ENV_GET_SYS_RESOURCES table function takes more than 1 second on AIX, HP, and Windows operating systems.

ENV_SYS_RESOURCES administrative view metadata

Table 247. ENV_SYS_RESOURCES table function metadata

| Column name | Data type | Description |
|----------------|---------------|--|
| NAME | VARCHAR(128) | Name of the attribute. See Table 248 for possible values.
Note: Some attributes might not be available depending on the operating system and hardware configuration at the server. |
| VALUE | VARCHAR(1024) | The value of the attribute. |
| DATATYPE | VARCHAR(128) | Attribute data type. |
| UNIT | VARCHAR(128) | Unit used for the VALUE column if applicable. NULL is returned if not applicable. |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Table 248. Possible values for the NAME column

| Information type | Name | Data Types | Description | Platforms that return this information | UNIT |
|------------------|-----------|--------------|--|--|------|
| Operating system | OS_NAME | VARCHAR(256) | Name of the operating system software. | All | NULL |
| Operating system | HOST_NAME | VARCHAR(256) | Host name of the system. | All | NULL |

Table 248. Possible values for the NAME column (continued)

| Information type | Name | Data Types | Description | Platforms that return this information | UNIT |
|------------------|------------------------|--------------|--|--|------|
| Operating system | OS_VERSION | VARCHAR(256) | Version of the operating system. For example, AIX: 4.3 version = 4. | All | NULL |
| Operating system | OS_RELEASE | VARCHAR(256) | Release of the operating system. For example, AIX: 4.3 release = 3. | All | NULL |
| Operating system | MACHINE_IDENTIFICATION | VARCHAR(256) | Machine hardware identification. | All | NULL |
| Operating system | OS_LEVEL | VARCHAR(256) | Maintenance level of the current version and release. For example, LINUX: 2.4.9, level = 9. | Linux | NULL |
| CPU | CPU_TOTAL | BIGINT | Total number of CPUs. | All | NULL |
| CPU | CPU_ONLINE | BIGINT | Number of CPUs online. | All | NULL |
| CPU | CPU_CONFIGURED | BIGINT | Number of CPUs configured. | All | NULL |
| CPU | CPU_SPEED | BIGINT | Speed of CPUs. | All | MHz |
| CPU | CPU_TIMEBASE | BIGINT | Frequency of timebase register increment. | Linux
PowerPC | Hz |
| CPU | CPU_HMT_DEGREE | BIGINT | On systems that support hardware multithreading (HMT), this is the number of processors that a physical processor will appear to the operating system as. On non-HMT systems, this value is 1. On HMT systems, "total" will reflect the number of logical CPUs. To get the number of physical CPUs, divide the "total" by "threadingDegree". | All | NULL |
| CPU | CPU_CORES_PER_SOCKET | BIGINT | Number of CPU cores per socket. On single core systems this value is 1. | All | NULL |
| Physical memory | MEMORY_TOTAL | BIGINT | Total size of physical memory. | All | MB |

Table 248. Possible values for the NAME column (continued)

| Information type | Name | Data Types | Description | Platforms that return this information | UNIT |
|------------------|----------------------|------------|--|--|---------|
| Physical memory | MEMORY_FREE | BIGINT | Amount of free physical memory. | All | MB |
| Physical memory | MEMORY_SWAP_TOTAL | BIGINT | Total amount of swap space. | All | MB |
| Physical memory | MEMORY_SWAP_FREE | BIGINT | Amount of free swap space. | All | MB |
| Virtual memory | VIRTUAL_MEM_TOTAL | BIGINT | Total amount of virtual memory on the system. | All | MB |
| Virtual memory | VIRTUAL_MEM_RESERVED | BIGINT | Amount of reserved virtual memory. | All | MB |
| Virtual memory | VIRTUAL_MEM_FREE | BIGINT | Amount of virtual memory free. | All | MB |
| CPU load | CPU_LOAD_SHORT | DECIMAL | Shortest period duration. For example, load samples over last 5 minutes. | All except Windows operating systems | NULL |
| CPU load | CPU_LOAD_MEDIUM | DECIMAL | Medium period duration. For example, load samples over last 10 minutes. | All except Windows operating systems | NULL |
| CPU load | CPU_LOAD_LONG | DECIMAL | Long period duration. For example, load samples over last 15 minutes. | All except Windows operating systems | NULL |
| CPU load | CPU_USAGE_TOTAL | SMALLINT | Percentage of overall CPU usage of the machine. | All | Percent |

GET_DB_CONFIG

The GET_DB_CONFIG procedure returns database configuration information.

The procedure does not take any arguments.

Note: This procedure has been deprecated and replaced by the “DBCFCG administrative view and DB_GET_CFG table function - Retrieve database configuration parameter information” on page 363.

►► GET_DB_CONFIG—(—) —————►►

The schema is SYSPROC.

The procedure returns a single result set with two rows containing a column for each parameter. The first column is named `DBCONFIG_TYPE`, as shown in the following table.

Table 249. Information returned by the `GET_DB_CONFIG` procedure

| Column name | Data type | Description |
|----------------------------|-----------|---|
| <code>DBCONFIG_TYPE</code> | INTEGER | The row with a value of 0 in this column contains the values of the database configuration parameters stored on disk. The row with a value of 1 in this column contains the current values of the database configuration parameters stored in memory. |

This procedure requires a user temporary table space that is used to create a global temporary table named `DB_CONFIG` to store the result set.

Authorization

One of the following authorities is required to execute the procedure:

- EXECUTE privilege on the procedure
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

Using the command line processor (CLP), change the value of the **logarchmeth1** database configuration parameter. Retrieve the original (on disk) and updated (in memory) values by calling the `GET_DB_CONFIG` procedure and then querying the resulting global temporary table (`DB_CONFIG`).

```
CONNECT TO SAMPLE

CREATE BUFFERPOOL MY8KPOOL SIZE 250 PAGESIZE 8K

CREATE USER TEMPORARY TABLESPACE MYTSP2 PAGESIZE
      8K MANAGED BY SYSTEM USING ( 'TSC2' ) BUFFERPOOL MY8KPOOL

UPDATE DB CFG USING LOGARCHMETH1 USEREXIT

CALL SYSPROC.GET_DB_CONFIG()

SELECT DBCONFIG_TYPE, LOGARCHMETH1
      FROM SESSION.DB_CONFIG

CONNECT RESET
```

The following is an example of output from this query.

| DBCONFIG_TYPE | LOGARCHMETH1 |
|---------------|--------------|
| 0 | 1 |
| 1 | 0 |

2 record(s) selected.

GET_DBM_CONFIG

The GET_DBM_CONFIG table function returns database manager configuration information.

The function does not take any arguments.

Note: This table function has been deprecated and replaced by the “DBMCFG administrative view and DBM_GET_CFG table function - Retrieve database manager configuration parameter information” on page 367.

►► GET_DBM_CONFIG (—) ◀◀

The schema is SYSFUN.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

The function returns a table with two rows containing a column for each parameter. The first column is named DBMCONFIG_TYPE, as shown in the following table.

Table 250. Information returned by the GET_DBM_CONFIG table function

| Column name | Data type | Description |
|----------------|-----------|---|
| DBMCONFIG_TYPE | INTEGER | The row with a value of 0 in this column contains the values of the database manager configuration parameters stored on disk. The row with a value of 1 in this column contains the current values of the database manager configuration parameters stored in memory. |

Example

Using the command line processor (CLP), change the value of the **numdb** and the **diaglevel** database manager configuration parameters, and then retrieve the original (on disk) and updated (in memory) values.

```

UPDATE DBM CFG USING NUMDB 32 DIAGLEVEL 4

CONNECT TO SAMPLE

SELECT DBMCONFIG_TYPE, NUMDB, DIAGLEVEL
FROM TABLE(SYSFUN.GET_DBM_CONFIG()) AS DBMCFG

CONNECT RESET

```

The following is an example of output from this query.

| DBMCONFIG_TYPE | NUMDB | DIAGLEVEL |
|----------------|-------|-----------|
| 0 | 32 | 4 |
| 1 | 8 | 3 |

2 record(s) selected.

Health snapshot routines

HEALTH_CONT_HI

The HEALTH_CONT_HI table function returns health indicator information for table space containers from a health snapshot of table spaces in a database.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

```

--> HEALTH_CONT_HI ( (dbname, member) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(255) that specifies a valid database name in the same instance as the currently connected database when calling this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify the null value to take the snapshot from the currently connected database.

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority

- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT * FROM TABLE(HEALTH_CONT_HI('',-1)) AS T
```

The following is an example of output from this query.

```

SNAPSHOT_TIMESTAMP      CONTAINER_NAME      ...
-----
2006-02-13-12.30.40.759542 D:\DB2\NODE0000\SAMPLE\T0000000\C0000000.CAT ...
2006-02-13-12.30.40.759542 D:\DB2\NODE0000\SAMPLE\T0000003\C0000000.LRG ...
2006-02-13-12.30.40.759542 D:\DB2\NODE0000\SAMPLE\T0000004\C0000000.UTM ...
2006-02-13-12.30.40.759542 D:\DB2\NODE0000\SAMPLE\T0000001\C0000000.TMP ...
2006-02-13-12.30.40.759542 D:\DB2\NODE0000\SAMPLE\T0000002\C0000000.LRG ...

```

5 record(s) selected.

Output from this query (continued).

```

... NODE_NUMBER HI_ID      HI_VALUE HI_TIMESTAMP      ...
... -----
...          -          3001          1 2006-02-13-12.26.26.158000 ...
...          -          3001          1 2006-02-13-12.26.26.158000 ...
...          -          3001          1 2006-02-13-12.26.26.158000 ...
...          -          3001          1 2006-02-13-12.26.26.158000 ...
...          -          3001          1 2006-02-13-12.26.26.158000 ...

```

Output from this query (continued).

```

... HI_ALERT_STATE      HI_ALERT_STATE_DETAIL HI_FORMULA      HI_ADDITIONAL_INFO
... -----
...          1 Normal          1          -
...          1 Normal          1          -
...          1 Normal          1          -
...          1 Normal          1          -
...          1 Normal          1          -

```

Information returned

Table 251. Information returned by the HEALTH_CONT_HI table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container name |
| NODE_NUMBER | INTEGER | node_number - Node number |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| HI_VALUE | SMALLINT | The value of the health indicator. |

Table 251. Information returned by the HEALTH_CONT_HI table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| HI_TIMESTAMP | TIMESTAMP | The date and time that the alert was generated. |
| HI_ALERT_STATE | BIGINT | The severity of the alert. |
| HI_ALERT_STATE_DETAIL | VARCHAR(20) | The text description of the HI_ALERT_STATE column. |
| HI_FORMULA | VARCHAR(2048) | The formula used to calculate the health indicator. |
| HI_ADDITIONAL_INFO | VARCHAR(4096) | Additional information about the health indicator. |

HEALTH_CONT_HI_HIS

Returns health indicator history information for containers from a health snapshot of a database.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►►—HEALTH_CONT_HI_HIS—(—dbname—,—member—)—————►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(255) that specifies a valid database name in the same instance as the currently connected database when calling this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify the null value to take the snapshot from the currently connected database.

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority

- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT * FROM TABLE(HEALTH_CONT_HI_HIS('','-1)) AS T
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | CONTAINER_NAME | ... |
|----------------------------|--|-----|
| 2006-02-13-12.30.41.915646 | D:\DB2\NODE0000\SAMPLE\T0000000\C0000000.CAT | ... |
| 2006-02-13-12.30.41.915646 | D:\DB2\NODE0000\SAMPLE\T0000000\C0000000.CAT | ... |
| 2006-02-13-12.30.41.915646 | D:\DB2\NODE0000\SAMPLE\T0000003\C0000000.LRG | ... |
| 2006-02-13-12.30.41.915646 | D:\DB2\NODE0000\SAMPLE\T0000003\C0000000.LRG | ... |
| 2006-02-13-12.30.41.915646 | D:\DB2\NODE0000\SAMPLE\T0000004\C0000000.UTM | ... |
| 2006-02-13-12.30.41.915646 | D:\DB2\NODE0000\SAMPLE\T0000004\C0000000.UTM | ... |
| 2006-02-13-12.30.41.915646 | D:\DB2\NODE0000\SAMPLE\T0000001\C0000000.TMP | ... |
| 2006-02-13-12.30.41.915646 | D:\DB2\NODE0000\SAMPLE\T0000001\C0000000.TMP | ... |
| 2006-02-13-12.30.41.915646 | D:\DB2\NODE0000\SAMPLE\T0000002\C0000000.LRG | ... |
| 2006-02-13-12.30.41.915646 | D:\DB2\NODE0000\SAMPLE\T0000002\C0000000.LRG | ... |

10 record(s) selected.

Output from this query (continued).

| ... | NODE_NUMBER | HI_ID | HI_TIMESTAMP | HI_VALUE | HI_ALERT_STATE | ... |
|-----|-------------|-------|----------------------------|----------|----------------|-----|
| ... | - | 3001 | 2006-02-13-12.16.25.911000 | 1 | 1 | ... |
| ... | - | 3001 | 2006-02-13-12.06.26.168000 | 1 | 1 | ... |
| ... | - | 3001 | 2006-02-13-12.16.25.911000 | 1 | 1 | ... |
| ... | - | 3001 | 2006-02-13-12.06.26.168000 | 1 | 1 | ... |
| ... | - | 3001 | 2006-02-13-12.16.25.911000 | 1 | 1 | ... |
| ... | - | 3001 | 2006-02-13-12.06.26.168000 | 1 | 1 | ... |
| ... | - | 3001 | 2006-02-13-12.16.25.911000 | 1 | 1 | ... |
| ... | - | 3001 | 2006-02-13-12.06.26.168000 | 1 | 1 | ... |
| ... | - | 3001 | 2006-02-13-12.16.25.911000 | 1 | 1 | ... |

Output from this query (continued).

| ... | HI_ALERT_STATE_DETAIL | HI_FORMULA | HI_ADDITIONAL_INFO |
|-----|-----------------------|------------|--------------------|
| ... | Normal | 1 | - |
| ... | Normal | 1 | - |
| ... | Normal | 1 | - |
| ... | Normal | 1 | - |
| ... | Normal | 1 | - |
| ... | Normal | 1 | - |
| ... | Normal | 1 | - |
| ... | Normal | 1 | - |
| ... | Normal | 1 | - |

Information returned

Table 252. Information returned by the HEALTH_CONT_HI_HIS table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container name |
| NODE_NUMBER | INTEGER | node_number - Node number |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| HI_TIMESTAMP | TIMESTAMP | The date and time that the alert was generated. |
| HI_VALUE | SMALLINT | The value of the health indicator. |
| HI_ALERT_STATE | BIGINT | The severity of the alert. |
| HI_ALERT_STATE_DETAIL | VARCHAR(20) | The text description of the HI_ALERT_STATE column. |
| HI_FORMULA | VARCHAR(2048) | The formula used to calculate the health indicator. |
| HI_ADDITIONAL_INFO | VARCHAR(4096) | Additional information about the health indicator. |

HEALTH_CONT_INFO

The HEALTH_CONT_INFO table function returns container information from a health snapshot of a database.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►►—HEALTH_CONT_INFO—(—dbname—,—member—)—►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(255) that specifies a valid database name in the same instance as the currently connected database when calling this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify the null value to take the snapshot from the currently connected database.

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT * FROM TABLE(HEALTH_CONT_INFO('',-1)) AS T
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | CONTAINER_NAME | |
|----------------------------|--|-----|
| 2006-02-13-12.30.40.541209 | D:\DB2\NODE0000\SAMPLE\T0000000\C0000000.CAT | ... |
| 2006-02-13-12.30.40.541209 | D:\DB2\NODE0000\SAMPLE\T0000003\C0000000.LRG | ... |
| 2006-02-13-12.30.40.541209 | D:\DB2\NODE0000\SAMPLE\T0000004\C0000000.UTM | ... |
| 2006-02-13-12.30.40.541209 | D:\DB2\NODE0000\SAMPLE\T0000001\C0000000.TMP | ... |
| 2006-02-13-12.30.40.541209 | D:\DB2\NODE0000\SAMPLE\T0000002\C0000000.LRG | ... |

5 record(s) selected.

Output from this query (continued).

| TABLESPACE_NAME | NODE_NUMBER | |
|------------------|-------------|-----|
| SYSCATSPACE | - | ... |
| SYSTOOLSPACE | - | ... |
| SYSTOOLSTMPSPACE | - | ... |
| TEMPSPACE1 | - | ... |
| USERSPACE1 | - | ... |

Output from this query (continued).

| ROLLED_UP_ALERT_STATE | ROLLED_UP_ALERT_STATE_DETAIL |
|-----------------------|------------------------------|
| 1 | Normal |
| 1 | Normal |
| 1 | Normal |
| 1 | Normal |
| 1 | Normal |

Information returned

Table 253. Information returned by the *HEALTH_CONT_INFO* table function

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container name |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table space name |
| NODE_NUMBER | INTEGER | node_number - Node number |
| ROLLED_UP_ALERT_STATE | BIGINT | The most severe alert state captured by this snapshot. |
| ROLLED_UP_ALERT_STATE_DETAIL | VARCHAR(20) | The text description of the ROLLED_UP_ALERT_STATE column. |

HEALTH_DB_HI

The *HEALTH_DB_HI* table function returns health indicator information from a health snapshot of a database.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►► *HEALTH_DB_HI* (—*dbname*—, —*member*—) ►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(255) that specifies a valid database name in the same instance as the currently connected database when calling this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify the null value to take the snapshot from all databases under the database instance.

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT * FROM TABLE(HEALTH_DB_HI('','-1)) AS T
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | HI_ID | DB_NAME | HI_VALUE | ... |
|----------------------------|-------|---------|----------|-----|
| 2006-02-13-12.30.23.949888 | 1001 | SAMPLE | 0 | ... |
| 2006-02-13-12.30.23.949888 | 1002 | SAMPLE | 0 | ... |
| 2006-02-13-12.30.23.949888 | 1003 | SAMPLE | 0 | ... |
| 2006-02-13-12.30.23.949888 | 1005 | SAMPLE | 6 | ... |
| 2006-02-13-12.30.23.949888 | 1006 | SAMPLE | 53 | ... |
| 2006-02-13-12.30.23.949888 | 1008 | SAMPLE | 3 | ... |
| 2006-02-13-12.30.23.949888 | 1010 | SAMPLE | 0 | ... |
| 2006-02-13-12.30.23.949888 | 1014 | SAMPLE | 74 | ... |
| 2006-02-13-12.30.23.949888 | 1015 | SAMPLE | 1 | ... |
| 2006-02-13-12.30.23.949888 | 1018 | SAMPLE | 1 | ... |
| 2006-02-13-12.30.23.949888 | 1022 | SAMPLE | 1 | ... |

11 record(s) selected.

Output from this query (continued).

| HI_TIMESTAMP | HI_ALERT_STATE | HI_ALERT_STATE_DETAIL | ... |
|----------------------------|----------------|-----------------------|-----|
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.30.25.640000 | 2 | Attention | ... |
| 2006-02-13-12.30.25.640000 | 2 | Attention | ... |
| 2006-02-13-12.29.25.281000 | 2 | Attention | ... |

Output from this query (continued).

| HI_FORMULA | ... |
|--------------------|-----|
| 0 | ... |
| ((0 / 5000) * 100) | ... |
| | ... |
| | ... |
| | ... |
| | ... |
| | ... |
| | ... |
| | ... |

```

... (((0 - 0) / ((118 - 0) + 1)) * 100)      ...
...                                           ...
...                                           ...
...                                           ...
...                                           ...
...                                           ...
... ((1170384 / (1170384 + 19229616)) * 100) ...
...                                           ...
...                                           ...
...                                           ...
...                                           ...
...                                           ...
... ((11155116032 / 21138935808) * 100)      ...
...                                           ...
...                                           ...
...                                           ...
...                                           ...
...                                           ...
... ((5264 / (50 * 4096)) * 100)              ...
... ((0 / 5) * 100)                          ...
... ((4587520 / 6160384) * 100)              ...
... -                                          ...
...                                           ...
...                                           ...
...                                           ...
...                                           ...
...                                           ...
... -                                          ...
...                                           ...
...                                           ...
...                                           ...
...                                           ...
...                                           ...
... -                                          ...
...                                           ...
...                                           ...
...                                           ...

```

Output from this query (continued).

```

... HI_ADDITIONAL_INFO
... -----
... -
... The high watermark for shared sort
... memory is "57". "99"% of the time
... the sort heap allocation is less
... than or equal to "246". The sort
... heap (sortheap) database
... configuration parameter is set
... to "256". The high watermark for
... private sort memory is "0".
... The sort heap (sortheap) database
... configuration parameter is set to
... "256". The high watermark for
... private sort memory is "57". The
... high watermark for shared sort
... memory is "0"
... The following are the related
... database configuration parameter
... settings: logprimary is "3",
... logsecond is "2", and logfilsiz
... is "1000". The application with
... the oldest transaction is "712".
... The following are the related

```

```

... database configuration parameter
... settings: logprimary is "3",
... logsecond is "2", and logfilsiz
... is "1000", blk_log_dsk_ful is
... "NO", logarchmeth1 is "OFF" and
... logarchmeth2 is "OFF".
... -
... -
... -
... The scope setting in the reorganization
... policy is "TABSCHEMA NOT LIKE 'SYS%'".
... Automatic reorganization (AUTO_REORG)
... for this database is set to "OFF".
... The longest estimated reorganization
... time is "N/A".
... The last successful backup was taken
... at "N/A". The log space consumed since
... this last backup has been "N/A" 4KB
... pages. Automation for database backup
... is set to "OFF". The last automated
... backup returned with SQLCODE = "N/A".
... The longest estimated backup time
... is "N/A".
... The scope is "N/A". Automatic
... statistics collection (AUTO_RUNSTATS)
... is set to "OFF".

```

Information returned

Table 254. Information returned by the HEALTH_DB_HI table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| HI_VALUE | SMALLINT | The value of the health indicator. |
| HI_TIMESTAMP | TIMESTAMP | The date and time that the alert was generated. |
| HI_ALERT_STATE | BIGINT | The severity of the alert. |
| HI_ALERT_STATE_DETAIL | VARCHAR(20) | The text description of the HI_ALERT_STATE column. |
| HI_FORMULA | VARCHAR(2048) | The formula used to calculate the health indicator. |
| HI_ADDITIONAL_INFO | VARCHAR(4096) | Additional information about the health indicator. |

HEALTH_DB_HI_HIS

The HEALTH_DB_HI_HIS table function returns health indicator history information from a health snapshot of a database.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is

not supported in Db2 pureScale environments. For more information, see "Health monitor has been deprecated" at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►►—HEALTH_DB_HI_HIS—(—*dbname*—,—*member*—)—————►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(255) that specifies a valid database name in the same instance as the currently connected database when calling this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify the null value to take the snapshot from all databases under the database instance.

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT * FROM TABLE(HEALTH_DB_HI_HIS('',-1)) AS T
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | HI_ID | DB_NAME | HI_VALUE | ... |
|----------------------------|-------|---------|----------|-----|
| 2006-02-13-12.30.26.325627 | 1001 | SAMPLE | 0 | ... |
| ... | | | | ... |
| 2006-02-13-12.30.26.325627 | 1002 | SAMPLE | 0 | ... |
| ... | | | | ... |
| 2006-02-13-12.30.26.325627 | 1003 | SAMPLE | 0 | ... |
| ... | | | | ... |
| 2006-02-13-12.30.26.325627 | 1005 | SAMPLE | 3 | ... |
| ... | | | | ... |
| 2006-02-13-12.30.26.325627 | 1008 | SAMPLE | 2 | ... |
| ... | | | | ... |

| | | |
|----------------------------|-------------|--------|
| 2006-02-13-12.30.26.325627 | 1010 SAMPLE | 0 ... |
| ... | | ... |
| 2006-02-13-12.30.26.325627 | 1014 SAMPLE | 73 ... |
| ... | | ... |
| 2006-02-13-12.30.26.325627 | 1015 SAMPLE | 1 ... |
| ... | | ... |
| 2006-02-13-12.30.26.325627 | 1018 SAMPLE | 1 ... |
| ... | | ... |
| 2006-02-13-12.30.26.325627 | 1022 SAMPLE | 1 ... |
| ... | | ... |

Output from this query (continued).

| ... HI_TIMESTAMP | HI_ALERT_STATE | HI_ALERT_STATE_DETAIL ... |
|--------------------------------|----------------|---------------------------|
| ... | ----- | ... |
| ... 2006-02-13-12.21.25.649000 | 1 Normal | ... |
| | | ... |
| ... 2006-02-13-12.21.25.649000 | 1 Normal | ... |
| | | ... |
| ... 2006-02-13-12.20.25.182000 | 1 Normal | ... |
| | | ... |
| ... 2006-02-13-12.16.25.911000 | 1 Normal | ... |
| | | ... |
| ... 2006-02-13-12.16.25.911000 | 1 Normal | ... |
| | | ... |
| ... 2006-02-13-12.16.25.911000 | 1 Normal | ... |
| | | ... |
| ... 2006-02-13-12.21.25.649000 | 1 Normal | ... |
| | | ... |
| ... 2006-02-13-12.29.55.461000 | 2 Attention | ... |
| | | ... |
| ... 2006-02-13-12.29.25.281000 | 2 Attention | ... |
| | | ... |
| ... 2006-02-13-12.27.55.743000 | 2 Attention | ... |
| | | ... |

Output from this query (continued).

| ... HI_FORMULA | ... |
|--|-------|
| ... | ----- |
| ... 0 | ... |
| | ... |
| ... ((0 / 5000) * 100) | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| | ... |
| ... (((0 - 0) / ((68 - 0) + 1)) * 100) | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| | ... |
| ... ((698410 / (698410 + 19701590)) * 100) | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| | ... |
| ... ((3920 / (50 * 4096)) * 100) | ... |
| ... | ... |
| ... | ... |
| ... ((0 / 4) * 100) | ... |

```

... HI_ADDITIONAL_INFO
... -----
... -
...
... The high watermark for shared sort
... memory is "15". "99%" of the time
... the sort heap allocation is less
... than or equal to "246". The sort
... heap (sortheap) database
... configuration parameter is set
... to "256". The high watermark
... for private sort memory is "0".
...
... The sort heap (sortheap) database
... configuration parameter is set
... to "256". The high watermark for
... private sort memory is "15". The
... high watermark for shared sort
... memory is "0"
...
... The following are the related
... database configuration parameter
... settings: logprimary is "3",
... logsecond is "2", and logfilsiz
... is "1000". The application with
... the oldest transaction is "712".
...
... -
...
... -
...
... -
...
... The scope setting in the
... reorganization policy is
... "TABSCHEMA NOT LIKE 'SYS%'".
... Automatic reorganization
... (AUTO_REORG) for this database
... is set to "OFF". The longest

```

```

... estimated reorganization time
... is "N/A".
...
... The last successful backup was taken
... at "N/A". The log space consumed
... since this last backup has been
... "N/A" 4KB pages. Automation for
... database backup is set to "OFF". The
... last automated backup returned with
... SQLCODE = "N/A". The longest
... estimated backup time is "N/A".
...
... The scope is "N/A". Automatic
... statistics collection
... (AUTO_RUNSTATS) is set to "OFF".
...

```

Information returned

Table 255. Information returned by the HEALTH_DB_HI_HIS table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| HI_VALUE | SMALLINT | The value of the health indicator. |
| HI_TIMESTAMP | TIMESTAMP | The date and time that the alert was generated. |
| HI_ALERT_STATE | BIGINT | The severity of the alert. |
| HI_ALERT_STATE_DETAIL | VARCHAR(20) | The text description of the HI_ALERT_STATE column. |
| HI_FORMULA | VARCHAR(2048) | The formula used to calculate the health indicator. |
| HI_ADDITIONAL_INFO | VARCHAR(4096) | Additional information about the health indicator. |

HEALTH_DB_HIC

The HEALTH_DB_HIC function returns collection health indicator information from a health snapshot of a database.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►►—HEALTH_DB_HIC—(—dbname—,—member—)—————◄◄

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(255) that specifies a valid database name in the same instance as the currently connected database when calling this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify the null value to take the snapshot from all databases under the database instance.

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT * FROM TABLE(HEALTH_DB_HIC('',-1)) AS T
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | HI_ID | DB_NAME | ... |
|----------------------------|-------|---------|-----|
| 2006-02-13-12.30.33.870959 | 1015 | SAMPLE | ... |
| 2006-02-13-12.30.33.870959 | 1022 | SAMPLE | ... |

2 record(s) selected.

Output from this query (continued).

| HI_OBJ_NAME | HI_OBJ_DETAIL | ... |
|---------------------------------------|---------------|-----|
| "JESSICAE"."EMPLOYEE" | REORG TABLE | ... |
| "SYSIBM"."SYSDATAPARTITIONEXPRESSION" | RUNSTATS | ... |

Output from this query (continued).

| HI_OBJ_STATE | HI_OBJ_STATE_DETAIL | HI_TIMESTAMP |
|--------------|---------------------|----------------------------|
| 2 | Attention | 2006-02-13-12.24.27.000000 |
| 2 | Attention | 2006-02-13-12.29.26.000000 |

Information returned

Table 256. Information returned by the `HEALTH_DB_HIC` table function

| Column name | Data type | Description or corresponding monitor element |
|---------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| HI_OBJ_NAME | VARCHAR(512) | A name that uniquely identifies an object in the collection. |
| HI_OBJ_DETAIL | VARCHAR(512) | Text that describes why the object was added to the collection. |
| HI_OBJ_STATE | SMALLINT | The state of the object. Valid states (defined in <code>sqlmon.h</code>) include: <ul style="list-style-type: none">• NORMAL (1). Action is not required on this object.• ATTENTION (2). Automation is not enabled for this health indicator; action must be taken manually.• AUTOMATED (5). Automation is enabled for this health indicator; action will be started automatically.• AUTOMATE_FAILED (6). Automation is enabled for this health indicator; action was started, but could not complete successfully. Manual intervention is now required. |
| HI_OBJ_STATE_DETAIL | VARCHAR(20) | A translated string version of the value in the <code>HI_OBJ_STATE</code> column. |
| HI_TIMESTAMP | TIMESTAMP | The date and time that the alert was generated. |

HEALTH_DB_HIC_HIS

Returns collection health indicator history information from a health snapshot of a database.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health

monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►►—HEALTH_DB_HIC_HIS—(—dbname—,—member—)—————►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(255) that specifies a valid database name in the same instance as the currently connected database when calling this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify the null value to take the snapshot from all databases under the database instance.

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for all active database partitions. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

SELECT * FROM TABLE(HEALTH_DB_HIC_HIS('',-1)) AS T

The following is an example of output from this query.

| HI_HIS_ENTRY_NUM | SNAPSHOT_TIMESTAMP | HI_ID | ... |
|------------------|----------------------------|-------|-----|
| 1 | 2006-02-13-12.30.34.496720 | 1015 | ... |
| 2 | 2006-02-13-12.30.34.496720 | 1022 | ... |
| 3 | 2006-02-13-12.30.34.496720 | 1022 | ... |
| 4 | 2006-02-13-12.30.34.496720 | 1022 | ... |
| 5 | 2006-02-13-12.30.34.496720 | 1022 | ... |
| 6 | 2006-02-13-12.30.34.496720 | 1022 | ... |
| 7 | 2006-02-13-12.30.34.496720 | 1022 | ... |
| 8 | 2006-02-13-12.30.34.496720 | 1022 | ... |

```

          9 2006-02-13-12.30.34.496720          1022 ...
         10 2006-02-13-12.30.34.496720          1022 ...

```

10 record(s) selected.

Output from this query (continued).

```

... DB_NAME      HI_OBJ_NAME      HI_OBJ_STATE ...
... -----
... SAMPLE      "JESSICAE"."EMPLOYEE"      2 ...
... SAMPLE      "SYSIBM"."SYSDATAPARTITIONEXPRESSION"      2 ...
... SAMPLE      "SYSIBM"."SYSDATAPARTITIONEXPRESSION"      2 ...
... SAMPLE      "SYSIBM"."SYSDATAPARTITIONEXPRESSION"      2 ...
... SAMPLE      "SYSIBM"."SYSDATAPARTITIONEXPRESSION"      1 ...
... SAMPLE      "SYSIBM"."SYSDATAPARTITIONEXPRESSION"      1 ...
... SAMPLE      "SYSIBM"."SYSDATAPARTITIONEXPRESSION"      1 ...
... SAMPLE      "SYSIBM"."SYSDATAPARTITIONEXPRESSION"      1 ...
... SAMPLE      "SYSIBM"."SYSDATAPARTITIONEXPRESSION"      1 ...
... SAMPLE      "SYSIBM"."SYSDATAPARTITIONEXPRESSION"      1 ...

```

Output from this query (continued).

```

... HI_OBJ_STATE_DETAIL  HI_TIMESTAMP
... -----
... Attention            2006-02-10-09.04.57.000000
... Attention            2006-02-13-12.27.56.000000
... Attention            2006-02-13-12.26.27.000000
... Attention            2006-02-13-12.24.56.000000
... Normal               2006-02-13-12.23.28.000000
... Normal               2006-02-13-12.21.56.000000
... Normal               2006-02-13-12.20.26.000000
... Normal               2006-02-13-12.18.57.000000
... Normal               2006-02-13-12.17.27.000000
... Normal               2006-02-13-12.15.56.000000

```

Information returned

Table 257. Information returned by the HEALTH_DB_HIC_HIS table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| HI_HIS_ENTRY_NUM | SMALLINT | A number that uniquely identifies the history entry. |
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| HI_OBJ_NAME | VARCHAR(512) | A name that uniquely identifies an object in the collection. |

Table 257. Information returned by the HEALTH_DB_HIC_HIS table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|-------------|---|
| HI_OBJ_STATE | SMALLINT | The state of the object. Valid states (defined in sqlmon.h) include: <ul style="list-style-type: none"> • NORMAL (1). Action is not required on this object. • ATTENTION (2). Automation is not enabled for this health indicator; action must be taken manually. • AUTOMATED (5). Automation is enabled for this health indicator; action will be started automatically. • AUTOMATE_FAILED (6). Automation is enabled for this health indicator; action was started, but could not complete successfully. Manual intervention is now required. |
| HI_OBJ_STATE_DETAIL | VARCHAR(20) | A translated string version of the value in the HI_OBJ_STATE column. |
| HI_TIMESTAMP | TIMESTAMP | The date and time that the alert was generated. |

HEALTH_DB_INFO

The HEALTH_DB_INFO table function returns information from a health snapshot of a database.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►►—HEALTH_DB_INFO—(—dbname—,—member—)—►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(255) that specifies a valid database name in the same instance as the currently connected database when calling

this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify the null value to take the snapshot from all databases under the database instance.

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT * FROM TABLE(HEALTH_DB_INFO(' ', -1)) AS T
```

The following is an example of output from this query.

```

SNAPSHOT_TIMESTAMP      DB_NAME      INPUT_DB_ALIAS      ...
-----
2006-02-13-12.30.23.340081 SAMPLE      SAMPLE      ...

```

1 record(s) selected.

Output from this query (continued).

```

... DB_PATH      DB_LOCATION SERVER_PLATFORM ...
... -----
... D:\DB2\NODE0000\SQL00003\      1      5 ...

```

Output from this query (continued).

```

... ROLLED_UP_ALERT_STATE ROLLED_UP_ALERT_STATE_DETAIL
... -----
...      4 Alarm

```

Information returned

Table 258. Information returned by the HEALTH_DB_INFO table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| DB_NAME | VARCHAR(128) | db_name - Database name |

Table 258. Information returned by the HEALTH_DB_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|---------------|---|
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input database alias |
| DB_PATH | VARCHAR(1024) | db_path - Database path |
| DB_LOCATION | INTEGER | db_location - Database location |
| SERVER_PLATFORM | INTEGER | server_platform - Server operating system |
| ROLLED_UP_ALERT_STATE | BIGINT | The most severe alert state captured by this snapshot. |
| ROLLED_UP_ALERT_STATE_DETAIL | VARCHAR(20) | The text description of the ROLLED_UP_ALERT_STATE column. |

HEALTH_DBM_HI

The HEALTH_DBM_HI table function returns health indicator information from a health snapshot of the database manager.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►►—HEALTH_DBM_HI—(—*member*—)—————►►

The schema is SYSPROC.

Table function parameter

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT * FROM TABLE(HEALTH_DBM_HI(-1)) AS T
```

The following is an example of output from this query.

```
SNAPSHOT_TIMESTAMP      HI_ID      SERVER_INSTANCE_NAME    ...
-----
2006-02-13-12.30.19.773632  1 DB2
2006-02-13-12.30.19.773632  4 DB2
```

2 record(s) selected.

Output from this query (continued).

```
... HI_VALUE HI_TIMESTAMP      HI_ALERT_STATE HI_ALERT_STATE_DETAIL ...
... -----
...      0 2006-02-13-12.26.26.158000      1 Normal
...     100 2006-02-13-12.26.26.158000      4 Alarm
```

Output from this query (continued).

```
... HI_FORMULA      HI_ADDITIONAL_INFO
... -----
...      0
... ((327680 / 327680) * 100)
```

Table 259. Information returned by the HEALTH_DBM_HI table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| SERVER_INSTANCE_NAME | VARCHAR(128) | server_instance_name - Server instance name |
| HI_VALUE | SMALLINT | The value of the health indicator. |
| HI_TIMESTAMP | TIMESTAMP | The date and time that the alert was generated. |
| HI_ALERT_STATE | BIGINT | The severity of the alert. |
| HI_ALERT_STATE_DETAIL | VARCHAR(20) | The text description of the HI_ALERT_STATE column. |
| HI_FORMULA | VARCHAR(2048) | The formula used to calculate the health indicator. |
| HI_ADDITIONAL_INFO | VARCHAR(4096) | Additional information about the health indicator. |

HEALTH_DBM_HI_HIS

The HEALTH_DBM_HI_HIS table function returns health indicator history information from a health snapshot of the database manager.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►►—HEALTH_DBM_HI_HIS—(—*member*—)—————►►

The schema is SYSPROC.

Table function parameter

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

SELECT * FROM TABLE(HEALTH_DBM_HI_HIS(-1)) AS T

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | HI_ID | SERVER_INSTANCE_NAME | HI_VALUE ... |
|----------------------------|-------|----------------------|--------------|
| 2006-02-13-12.30.20.460905 | 1 | DB2 | 0 ... |
| 2006-02-13-12.30.20.460905 | 1 | DB2 | 0 ... |
| 2006-02-13-12.30.20.460905 | 1 | DB2 | 0 ... |
| 2006-02-13-12.30.20.460905 | 1 | DB2 | 0 ... |
| 2006-02-13-12.30.20.460905 | 1 | DB2 | 0 ... |
| 2006-02-13-12.30.20.460905 | 1 | DB2 | 0 ... |
| 2006-02-13-12.30.20.460905 | 1 | DB2 | 0 ... |
| 2006-02-13-12.30.20.460905 | 1 | DB2 | 0 ... |
| 2006-02-13-12.30.20.460905 | 1 | DB2 | 0 ... |
| 2006-02-13-12.30.20.460905 | 4 | DB2 | 100 ... |
| 2006-02-13-12.30.20.460905 | 4 | DB2 | 100 ... |
| 2006-02-13-12.30.20.460905 | 4 | DB2 | 100 ... |
| 2006-02-13-12.30.20.460905 | 4 | DB2 | 100 ... |
| 2006-02-13-12.30.20.460905 | 4 | DB2 | 60 ... |
| 2006-02-13-12.30.20.460905 | 4 | DB2 | 60 ... |
| 2006-02-13-12.30.20.460905 | 4 | DB2 | 60 ... |

```

2006-02-13-12.30.20.460905      4 DB2      60 ...
2006-02-13-12.30.20.460905      4 DB2      60 ...

```

18 record(s) selected.

Output for this query (continued).

```

... HI_TIMESTAMP      HI_ALERT_STATE HI_ALERT_STATE_DETAIL ...
... -----
... 2006-02-13-12.21.25.649000      1 Normal      ...
... 2006-02-13-12.16.25.911000      1 Normal      ...
... 2006-02-13-12.11.25.377000      1 Normal      ...
... 2006-02-13-12.06.26.168000      1 Normal      ...
... 2006-02-13-12.01.25.165000      1 Normal      ...
... 2006-02-13-11.56.25.927000      1 Normal      ...
... 2006-02-13-11.51.25.452000      1 Normal      ...
... 2006-02-13-11.46.25.211000      1 Normal      ...
... 2006-02-13-11.41.25.972000      1 Normal      ...
... 2006-02-13-12.21.25.649000      4 Alarm       ...
... 2006-02-13-12.16.25.911000      4 Alarm       ...
... 2006-02-13-12.11.25.377000      4 Alarm       ...
... 2006-02-13-12.06.26.168000      4 Alarm       ...
... 2006-02-13-12.01.25.165000      1 Normal      ...
... 2006-02-13-11.56.25.927000      1 Normal      ...
... 2006-02-13-11.51.25.452000      1 Normal      ...
... 2006-02-13-11.46.25.211000      1 Normal      ...
... 2006-02-13-11.41.25.972000      1 Normal      ...

```

Output for this query (continued).

```

... HI_FORMULA      HI_ADDITIONAL_INFO
... -----
... 0      -
... 0      -
... 0      -
... 0      -
... 0      -
... 0      -
... 0      -
... 0      -
... 0      -
... 0      -
... ((327680 / 327680) * 100)      -
... ((327680 / 327680) * 100)      -
... ((327680 / 327680) * 100)      -
... ((327680 / 327680) * 100)      -
... ((196608 / 327680) * 100)      -
... ((196608 / 327680) * 100)      -
... ((196608 / 327680) * 100)      -
... ((196608 / 327680) * 100)      -
... ((196608 / 327680) * 100)      -
... ((196608 / 327680) * 100)      -

```

Information returned

Table 260. Information returned by the HEALTH_DBM_HI_HIS table function

| Column name | Data type | Description or corresponding monitor element |
|----------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| SERVER_INSTANCE_NAME | VARCHAR(128) | server_instance_name - Server instance name |

Table 260. Information returned by the HEALTH_DBM_HI_HIS table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| HI_VALUE | SMALLINT | The value of the health indicator. |
| HI_TIMESTAMP | TIMESTAMP | The date and time that the alert was generated. |
| HI_ALERT_STATE | BIGINT | The severity of the alert. |
| HI_ALERT_STATE_DETAIL | VARCHAR(20) | The text description of the HI_ALERT_STATE column. |
| HI_FORMULA | VARCHAR(2048) | The formula used to calculate the health indicator. |
| HI_ADDITIONAL_INFO | VARCHAR(4096) | Additional information about the health indicator. |

HEALTH_DBM_INFO

The HEALTH_DBM_INFO function returns information from a health snapshot of the database manager.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►►—HEALTH_DBM_INFO—(—*member*—)—————►

The schema is SYSPROC.

Table function parameter

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT * FROM TABLE(HEALTH_DBM_INFO(-1)) AS T
```

The following is an example of output from this query.

```
SNAPSHOT_TIMESTAMP      SERVER_INSTANCE_NAME      ROLLED_UP_ALERT_STATE    ...
-----
2006-02-13-12.30.19.663924 DB2                                4    ...
```

1 record(s) selected.

Output from this query (continued).

```
... ROLLED_UP_ALERT_STATE_DETAIL DB2START_TIME    ...
... -----
... Alarm                                2006-02-09-10.56.18.126182 ...
```

Output from this query (continued).

```
... LAST_RESET          NUM_NODES_IN_DB2_INSTANCE
... -----
... -                                1
```

Information returned

Table 261. Information returned by the HEALTH_DBM_INFO table function

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| SERVER_INSTANCE_NAME | VARCHAR(128) | server_instance_name - Server instance name |
| ROLLED_UP_ALERT_STATE | BIGINT | The most severe alert state captured by this snapshot. |
| ROLLED_UP_ALERT_STATE_DETAIL | VARCHAR(20) | The text description of the ROLLED_UP_ALERT_STATE column. |
| DB2START_TIME | TIMESTAMP | db2start_time - Start database manager timestamp |
| LAST_RESET | TIMESTAMP | last_reset - Last reset timestamp |
| NUM_NODES_IN_DB2_INSTANCE | INTEGER | num_nodes_in_db2_instance - Number of nodes in database partition |

HEALTH_GET_ALERT_ACTION_CFG

Returns health alert action configuration settings for various object types (database manager, database, table space, and table space container) and for various configuration levels (install default, instance, global, and object).

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

```
►►—HEALTH_GET_ALERT_ACTION_CFG—(—objecttype—,—cfg_level—,—dbname—,—————►
►—objectname—)——————►◄◄
```

The schema is SYSPROC.

Table function parameters

objecttype

An input argument of type VARCHAR(3) that indicates the object type. The value must be one of the following case-insensitive values:

- 'DBM' for database manager
- 'DB' for database
- 'TS' for table space
- 'TSC' for table space container

Note: Leading and trailing spaces will be ignored.

cfg_level

An input argument of type VARCHAR(1) that indicates the configuration level. The value must be one of the following case-insensitive values:

- For *objecttype* 'DBM': 'D' for install default; 'G' or 'O' for instance level.
- For *objecttype* that is not 'DBM': 'D' for install default; 'G' for global level; 'O' for object level.

dbname

An input argument of type VARCHAR(128) that indicates the database name. The database name must be provided if *objecttype* is 'DB', 'TS', or 'TSC', and *cfg_level* is 'O'. For all other combinations of *objecttype* and *cfg_level*, the *dbname* parameter should be NULL (or an empty string).

objectname

An input argument of type VARCHAR(1024) that indicates the object name, for example, <table space name> or <table space name>.<container name>. The object name must be provided if *objecttype* is 'TS' or 'TSC', and *cfg_level* is 'O'. For all other combinations of *objecttype* and *cfg_level*, the *objectname* parameter should be NULL (or an empty string).

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Retrieve object level alert action configuration settings for database SAMPLE for health indicator ID 1004.

```
SELECT OBJECTTYPE, CFG_LEVEL, SUBSTR(DBNAME,1,8) AS DBNAME,
       SUBSTR(OBJECTNAME,1,8) AS OBJECTNAME, ID, IS_DEFAULT,
       SUBSTR(CONDITION,1,10) AS CONDITION, ACTIONTYPE,
       SUBSTR(ACTIONNAME,1,30) AS ACTIONNAME, SUBSTR(USERID,1,8) AS USERID,
       SUBSTR(HOSTNAME,1,10) AS HOSTNAME, SCRIPT_TYPE,
       SUBSTR(WORKING_DIR,1,10) AS WORKING_DIR, TERMINATION_CHAR,
       SUBSTR(PARAMETERS,1,10) AS PARAMETERS
FROM TABLE(HEALTH_GET_ALERT_ACTION_CFG('DB','0','SAMPLE','')) AS ACTION_CFG
WHERE ID = 1004
```

The following is an example of output for this query.

| OBJECTTYPE | CFG_LEVEL | DBNAME | OBJECTNAME | ID | IS_DEFAULT | CONDITION |
|------------|-----------|--------|------------|------|------------|-----------|
| DB | 0 | SAMPLE | | 1004 | 1 | ALARM |
| DB | 0 | SAMPLE | | 1004 | 1 | ALARM |

2 record(s) selected.

Output for this query (continued).

| ... | ACTIONTYPE | ACTIONNAME | USERID | HOSTNAME |
|-----|------------|-------------------------------|--------|----------|
| ... | S | ~/health_center/script/scrpn6 | uid1 | - |
| ... | T | 00.0005 | uid1 | HOST3 |

Output for this query (continued).

| ... | SCRIPT_TYPE | WORKING_DIR | TERMINATION_CHAR | PARAMETERS |
|-----|-------------|-------------|------------------|------------|
| ... | 0 | ~/health_c | - | - |
| ... | - | - | - | - |

Example 2: Retrieve the condition, action type, action name, hostname, and script type for database SAMPLE for health indicator ID 1004.

```
SELECT CONDITION, ACTIONTYPE, SUBSTR(ACTIONNAME,1,35) AS ACTIONNAME,
       SUBSTR(USERID,1,8) AS USERID, SUBSTR(HOSTNAME,1,10) AS HOSTNAME, SCRIPT_TYPE
FROM TABLE(HEALTH_GET_ALERT_ACTION_CFG('DB','0','SAMPLE','')) AS ALERT_ACTION_CFG
WHERE ID=1004
```

The following is an example of output for this query.

| CONDITION | ACTIONTYPE | ACTIONNAME | ... |
|-----------|------------|-------------------------------|-----|
| ALARM | S | ~/health_center/script/scrpn6 | ... |
| ALARM | T | 00.0005 | ... |

2 record(s) selected.

Output for this query (continued).

| ... | USERID | HOSTNAME | SCRIPT_TYPE |
|-----|--------|----------|-------------|
| ... | uid1 | - | 0 |
| ... | uid1 | HOST3 | - |

Usage notes

The HEALTH_GET_IND_DEFINITION table function can be used to map health indicator IDs to the health indicator names.

Information returned

Table 262. Information returned by the HEALTH_GET_ALERT_ACTION_CFG table function

| Column name | Data type | Description |
|------------------|---------------|---|
| OBJECTTYPE | VARCHAR(3) | Object type. |
| CFG_LEVEL | CHAR(1) | Configuration level. |
| DBNAME | VARCHAR(128) | Database name. |
| OBJECTNAME | VARCHAR(512) | object_name - Object name monitor element |
| ID | BIGINT | id - cluster caching facility identification monitor element |
| IS_DEFAULT | SMALLINT | Whether the settings is the default: 1 if it is the default, 0 if it is not the default, Null if it is not applicable. |
| CONDITION | VARCHAR(32) | Alert condition upon which the action is triggered. |
| ACTIONTYPE | CHAR(1) | Action type: 'S' for script action or 'T' for task action. |
| ACTIONNAME | VARCHAR(5000) | If ACTIONTYPE is 'S', this is the script path name. If ACTIONTYPE is 'T', this is the task ID. |
| USERID | VARCHAR(1024) | User name under which the action will be executed. |
| HOSTNAME | VARCHAR(255) | hostname - Host name monitor element |
| SCRIPT_TYPE | CHAR(1) | Script type: If ACTIONTYPE is 'S', 'O' for an operating system command script or 'D' for a database command script; if ACTIONTYPE is 'T', this value is null. |
| WORKING_DIR | VARCHAR(5000) | If ACTIONTYPE is 'S', the working directory for the script; if ACTIONTYPE is 'T', this value is null. |
| TERMINATION_CHAR | VARCHAR(4) | For a database command script action, the statement termination character; otherwise, this values is null. |
| PARAMETERS | VARCHAR(200) | For a operating system command script action, the command line parameters. |

HEALTH_GET_ALERT_CFG

Returns health alert configuration settings for various object types (database manager, database, table space, table space container) and for various configuration levels (install default, global, and object).

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

```
►►HEALTH_GET_ALERT_CFG(—objecttype—,—cfg_level—,—dbname—,——————►  
►—objectname—)—————►◄
```

The schema is SYSPROC.

Table function parameters

objecttype

An input argument of type VARCHAR(3) that indicates the object type. The value must be one of the following case-insensitive values:

- 'DBM' for database manager
- 'DB' for database
- 'TS' for table space
- 'TSC' for table space container

Note: Leading and trailing spaces will be ignored.

cfg_level

An input argument of type VARCHAR(1) that indicates the configuration level. The value must be one of the following case-insensitive values:

- For *objecttype* 'DBM': 'D' for install default; 'G' or 'O' for instance level.
- For *objecttype* that is not 'DBM': 'D' for install default; 'G' for global level; 'O' for object level.

dbname

An input argument of type VARCHAR(128) that indicates the database name. The database name must be provided if *objecttype* is 'DB', 'TS', or 'TSC', and *cfg_level* is 'O'. For all other combinations of *objecttype* and *cfg_level*, the *dbname* parameter should be NULL (or an empty string).

objectname

An input argument of type VARCHAR(1024) that indicates the object name, for example, <table space name> or <table space name>.<container name>. The object name must be provided if *objecttype* is 'TS' or 'TSC', and *cfg_level* is 'O'. For all other combinations of *objecttype* and *cfg_level*, the *objectname* parameter should be NULL (or an empty string).

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine

- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Retrieve the object level alert configuration settings for database SAMPLE.

```
SELECT * FROM TABLE(SYSPROC.HEALTH_GET_ALERT_CFG('DB','0','SAMPLE',''))
      AS ALERT_CFG
```

The following is an example of output for this query.

| OBJECTTYPE | CFG_LEVEL | DBNAME | OBJECTNAME | ... |
|------------|-----------|--------|------------|-----|
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| DB | 0 | SAMPLE | | ... |
| ... | | | | ... |

Output for this query (continued).

| ... | ID | IS_DEFAULT | WARNING_THRESHOLD | ... |
|-----|------|------------|-------------------|-----|
| ... | | | | ... |
| ... | 1001 | 0 | 0 | ... |
| ... | 1018 | 0 | 0 | ... |
| ... | 1015 | 0 | 0 | ... |
| ... | 1022 | 0 | 0 | ... |
| ... | 1002 | 1 | 95 | ... |
| ... | 1003 | 1 | 30 | ... |
| ... | 1004 | 1 | 60 | ... |
| ... | 1005 | 1 | 75 | ... |
| ... | 1006 | 1 | 75 | ... |
| ... | 1007 | 1 | 5 | ... |
| ... | 1008 | 1 | 75 | ... |
| ... | 1009 | 1 | 5 | ... |
| ... | 1010 | 1 | 50 | ... |
| ... | 1011 | 1 | 80 | ... |

Output for this query (continued).

| ... | ALARM_THRESHOLD | SENSITIVITY | EVALUATE | ACTION_ENABLED | ... |
|-----|-----------------|-------------|----------|----------------|-----|
| ... | | | | | ... |
| ... | 0 | | 0 | 0 | 0 |
| ... | 0 | | 0 | 1 | 0 |
| ... | 0 | | 0 | 1 | 0 |
| ... | 0 | | 0 | 1 | 0 |
| ... | 100 | | 0 | 0 | 0 |

| | | | | |
|-----|----|---|---|---|
| ... | 50 | 0 | 1 | 0 |
| ... | 30 | 0 | 1 | 0 |
| ... | 85 | 0 | 1 | 0 |
| ... | 85 | 0 | 1 | 0 |
| ... | 10 | 0 | 1 | 0 |
| ... | 85 | 0 | 1 | 0 |
| ... | 10 | 0 | 1 | 0 |
| ... | 70 | 0 | 1 | 0 |
| ... | 70 | 0 | 0 | 0 |

Example 2: Retrieve the warning and alarm thresholds for the health indicator ID '2002' for table space USERSPACE1 in database SAMPLE.

```
SELECT WARNING_THRESHOLD, ALARM_THRESHOLD
  FROM TABLE(SYSPROC.HEALTH_GET_ALERT_CFG('TS','0','SAMPLE','USERSPACE1'))
 AS T WHERE ID = 2002
```

The following is an example of output for this query.

| WARNING_THRESHOLD | ALARM_THRESHOLD |
|-------------------|-----------------|
| 80 | 90 |

SQL22004N Cannot find the requested configuration for the given object.
Returning default configuration for "tablespaces".

1 record(s) selected with 1 warning messages printed.

Usage notes

The HEALTH_GET_IND_DEFINITION table function can be used to map health indicator IDs to the health indicator names.

Example: Retrieve the warning and alarm thresholds for the health indicator Tablespace Utilization (ts.ts_util) for table space USERSPACE1 in database SAMPLE.

```
WITH HINAME(ID) AS (SELECT ID FROM TABLE(SYSPROC.HEALTH_GET_IND_DEFINITION('')) AS W
 WHERE NAME = 'ts.ts_util')
SELECT WARNING_THRESHOLD, ALARM_THRESHOLD
  FROM TABLE(SYSPROC.HEALTH_GET_ALERT_CFG('TS','0','SAMPLE','USERSPACE1')) AS T,
 HINAME AS H
 WHERE T.ID = H.ID
```

The following is an example of output for this query.

| WARNING_THRESHOLD | ALARM_THRESHOLD |
|-------------------|-----------------|
| 80 | 90 |

SQL22004N Cannot find the requested configuration for the given object.
Returning default configuration for "tablespaces".

1 record(s) selected with 1 warning messages printed.

Information returned

Table 263. Information returned by the HEALTH_GET_ALERT_CFG table function

| Column name | Data type | Description |
|-------------|--------------|----------------------|
| OBJECTTYPE | VARCHAR(3) | Object type. |
| CFG_LEVEL | CHAR(1) | Configuration level. |
| DBNAME | VARCHAR(128) | Database name. |

Table 263. Information returned by the HEALTH_GET_ALERT_CFG table function (continued)

| Column name | Data type | Description |
|-------------------|--------------|---|
| OBJECTNAME | VARCHAR(512) | object_name - Object name monitor element |
| ID | BIGINT | id - cluster caching facility identification monitor element |
| IS_DEFAULT | SMALLINT | Whether the settings is the default: 1 if it is the default, 0 if it is not the default or Null if not applicable. |
| WARNING_THRESHOLD | BIGINT | Warning threshold. Null if not applicable. |
| ALARM_THRESHOLD | BIGINT | Alarm threshold. Null if not applicable. |
| SENSITIVITY | BIGINT | Health indicator sensitivity. |
| EVALUATE | SMALLINT | 1 if this health indicator is being evaluated or 0 if it is not being evaluated. |
| ACTION_ENABLED | SMALLINT | 1 if an action is enabled to run upon an alert occurrence or 0 if no action is enabled to run upon an alert occurrence. |

HEALTH_GET_IND_DEFINITION

Returns the health indicator definitions.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►►—HEALTH_GET_IND_DEFINITION—(—*locale*—)—————◄◄

The schema is SYSPROC.

Table function parameter

locale

An input argument of type VARCHAR(33) that indicates the locale in which the translatable output is to be returned. If the input locale is not supported by the database server, an SQL warning message is issued, and the default language (English) is used. If the input locale is not provided, that is, its value is NULL (or an empty string), the default language is used.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Retrieve the type and short description for health indicator db.db_op_status in French.

```
SELECT TYPE, SHORT_DESCRIPTION
  FROM TABLE(SYSPROC.HEALTH_GET_IND_DEFINITION('fr_FR'))
        AS IND_DEFINITION WHERE NAME = 'db.db_op_status'
```

The following is an example of output for this query.

| TYPE | SHORT_DESCRIPTION |
|-------|---|
| STATE | Etat opérationnel de la base de données |

1 record(s) selected.

Example 2: Retrieve the short description for health indicator ID 1001 in English.

```
SELECT SHORT_DESCRIPTION FROM TABLE(SYSPROC.HEALTH_GET_IND_DEFINITION('en_US')
        AS IND_DEFINITION WHERE ID = 1001
```

The following is an example of output for this query.

| SHORT_DESCRIPTION |
|----------------------------|
| Database Operational State |

Example 3: Retrieve all health indicator IDs and names.

```
SELECT ID, NAME FROM TABLE(HEALTH_GET_IND_DEFINITION('')) AS T
```

The following is an example of output for this query.

| ID | NAME |
|------|-----------------------|
| 1 | db2.db2_op_status |
| 2 | db2.sort_privmem_util |
| 4 | db2.mon_heap_util |
| 1001 | db.db_op_status |
| 1002 | db.sort_shrmem_util |
| ... | |
| 2001 | ts.ts_op_status |
| 2002 | ts.ts_util |
| ... | |
| 3002 | tsc.tscont_util |
| 1015 | db.tb_reorg_req |
| ... | |

Information returned

Table 264. Information returned by the *HEALTH_GET_IND_DEFINITION* table function

| Column name | Data type | Description |
|-------------------|----------------|---|
| ID | BIGINT | id - cluster caching facility identification monitor element |
| NAME | VARCHAR(128) | Health indicator name. |
| SHORT_DESCRIPTION | VARCHAR(1024) | Health indicator short description. |
| LONG_DESCRIPTION | VARCHAR(32672) | Health indicator long description. |
| TYPE | VARCHAR(16) | Health indicator type. Possible values are: <ul style="list-style-type: none">• 'THRESHOLD_UPPER': upper-bounded threshold-based health indicators.• 'THRESHOLD_LOWER': lower-bounded threshold-based health indicators.• 'STATE': state-based health indicators.• 'COLLECTION_STATE': collection state-based health indicators. |
| UNIT | VARCHAR(1024) | Unit of the health indicator values and thresholds or Null if not applicable. |
| CATEGORY | VARCHAR(1024) | Health indicator category. |
| FORMULA | VARCHAR(512) | Health indicator formula. |
| REFRESH_INTERVAL | BIGINT | Health indicator evaluation interval in seconds. |

HEALTH_HI_REC

Retrieves a set of recommendations that address a health indicator in alert state on a particular database object. Recommendations are returned in an XML document that contains information about actions that can be taken (for example, scripts that can be run) to resolve the alert state.

Important: This procedure has been deprecated and might be removed in a future release because the health monitor was deprecated in Version 9.7.

Syntax

```
►►—HEALTH_HI_REC—(—schema-version—,—indicator-id—,—dbname—,——————►  
—object-type—,—object-name—,—dbpartitionnum—,—client-locale—,——————►  
►—recommendation-doc—)—————►◄
```

The schema is SYSPROC.

Any scripts that are returned by this procedure must be invoked from the instance on which the health indicator entered the alert state.

If the specified health indicator on the identified object is not in an alert state, an error is returned (SQLSTATE 5U0ZZ).

Procedure parameters

schema-version

An input argument of type INTEGER that specifies the version ID of the schema used to represent the XML document. The recommendation document will only contain elements and attributes that were defined for that schema version. Valid schema versions are defined in `db2ApiDf.h`, located in the include subdirectory of the `sqllib` directory.

indicator-id

An input argument of type INTEGER that specifies the numeric identifier of the health indicator for which recommendations are being requested. Valid health indicator IDs are defined in `sqlmon.h`, located in the include subdirectory of the `sqllib` directory.

dbname

An input argument of type VARCHAR(255) that specifies an alias name for the database against which the health indicator entered an alert state, and when object type is either `DB2HEALTH_OBJTYPE_TS_CONTAINER`, `DB2HEALTH_OBJTYPE_TABLESPACE`, or `DB2HEALTH_OBJTYPE_DATABASE`. Specify NULL otherwise.

object-type

An input argument of type INTEGER that specifies the type of object on which the health indicator entered an alert state. Valid object types are defined in `sqlmon.h`, located in the include subdirectory of the `sqllib` directory.

object-name

An input argument of type VARCHAR(255) that specifies the name of a table space or table space container when the object type is set to `DB2HEALTH_OBJTYPE_TABLESPACE` or `DB2HEALTH_OBJTYPE_TS_CONTAINER`. Specify NULL if the object type is `DB2HEALTH_OBJTYPE_DATABASE` or `DB2HEALTH_OBJTYPE_DATABASE_MANAGER`. In the case of a table space container, the object name is specified as *table_space_name.container_name*.

dbpartitionnum

An input argument of type INTEGER that specifies the number of the database partition on which the health indicator entered an alert state. Valid values are 0 to 999, -1 (which specifies the currently connected database partition), and -2 (which specifies all active database partitions). An active database partition is a partition where the database is available for connection and use by applications.

client-locale

An input argument of type VARCHAR(33) that specifies a client language identifier. Use this parameter to specify the language in which recommendations are to be returned. If no value is specified, 'En_US' (English) will be used. Note that if the message files for the specified locale are not available on the server, 'En_US' will be used as the default.

recommendation-doc

An output argument of type BLOB(2M) that contains the recommendation

document (XML), formatted according to the Health Recommendation schema definition (see the XML schema DB2RecommendationSchema.xsd, located in the misc subdirectory of the sql11b directory). The XML document is encoded in UTF-8, and text in the document is in the locale of the caller, or English, if messages are not available in the caller's locale at the target instance.

Authorization

One of the following authorities is required to execute the procedure:

- EXECUTE privilege on the procedure
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

HEALTH_TBS_HI

Returns health indicator information for table spaces from a health snapshot of table spaces in a database.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see "Health monitor has been deprecated" at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►►—HEALTH_TBS_HI—(—*dbname*—,—*member*—)——————►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(255) that specifies a valid database name in the same instance as the currently connected database when calling this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify the null value to take the snapshot from the currently connected database.

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT * FROM TABLE(HEALTH_TBS_HI('','-1)) AS T
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | TABLESPACE_NAME | HI_ID | HI_VALUE | ... |
|----------------------------|------------------|-------|----------|-----|
| 2006-02-13-12.30.35.229196 | SYSCATSPACE | 2001 | 0 | ... |
| 2006-02-13-12.30.35.229196 | SYSCATSPACE | 2002 | 99 | ... |
| 2006-02-13-12.30.35.229196 | SYSCATSPACE | 2003 | 0 | ... |
| 2006-02-13-12.30.35.229196 | SYSTOOLSPACE | 2001 | 0 | ... |
| 2006-02-13-12.30.35.229196 | SYSTOOLSPACE | 2002 | 62 | ... |
| 2006-02-13-12.30.35.229196 | SYSTOOLSPACE | 2003 | 0 | ... |
| 2006-02-13-12.30.35.229196 | SYSTOOLSTMPSPACE | 2001 | 0 | ... |
| 2006-02-13-12.30.35.229196 | TEMPSPACE1 | 2001 | 0 | ... |
| 2006-02-13-12.30.35.229196 | USERSPACE1 | 2001 | 0 | ... |
| 2006-02-13-12.30.35.229196 | USERSPACE1 | 2002 | 100 | ... |
| 2006-02-13-12.30.35.229196 | USERSPACE1 | 2003 | 0 | ... |

11 record(s) selected.

Output from this query (continued).

| HI_TIMESTAMP | HI_ALERT_STATE | HI_ALERT_STATE_DETAIL | ... |
|----------------------------|----------------|-----------------------|-----|
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 4 | Alarm | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |
| 2006-02-13-12.26.26.158000 | 4 | Alarm | ... |
| 2006-02-13-12.26.26.158000 | 1 | Normal | ... |

Output from this query (continued).

| HI_FORMULA | HI_ADDITIONAL_INFO |
|-----------------------|--|
| 0 | - |
| ((9376 / 9468) * 100) | The short term table space growth rate from "02/13/2006 11:26:26.000158" to "02/13/2006 12:26:26.000158" is "N/A" bytes per second and the long term growth rate from "02/12/2006 12:26:26.000158" to "02/13/2006 12:26:26.000158" is "N/A" bytes per second. Time to fullness is projected to be "N/A" and "N/A". |

| | | |
|-----|-----------------------|--|
| ... | 0 | The table space is defined with automatic storage set to "YES" and automatic resize enabled set to "YES". |
| ... | 0 | The table space is defined with automatic storage set to "YES" and automatic resize enabled set to "YES". The following are the automatic resize settings: increase size (bytes) "-1", increase size (percent) "N/A", maximum size (bytes) "-1". The current table space size (bytes) is "38797312". |
| ... | 0 | - |
| ... | ((156 / 252) * 100) | The short term table space growth rate from "02/13/2006 11:26:26.000158" to "02/13/2006 12:26:26.000158" is "N/A" bytes per second and the long term growth rate from "02/12/2006 12:26:26.000158" to "02/13/2006 12:26:26.000158" is "N/A" bytes per second. Time to fullness is projected to be "N/A" and "N/A". |
| ... | 0 | The table space is defined with automatic storage set to "YES" and automatic resize enabled set to "YES". |
| ... | 0 | The table space is defined with automatic storage set to "YES" and automatic resize enabled set to "YES". The following are the automatic resize settings: increase size (bytes) "-1", increase size (percent) "N/A", maximum size (bytes) "-1". The current table space size (bytes) is "1048576". |
| ... | 0 | - |
| ... | 0 | - |
| ... | 0 | - |
| ... | ((1504 / 1504) * 100) | The short term table space growth rate from "02/13/2006 11:26:26.000158" to "02/13/2006 12:26:26.000158" is "N/A" bytes per second and the long term growth rate from "02/12/2006 12:26:26.000158" to "02/13/2006 12:26:26.000158" is "N/A" bytes per second. Time to fullness is projected to be "N/A" and "N/A". The table space is defined with automatic storage set to "YES" and automatic resize enabled set to "YES". |
| ... | 0 | The table space is defined with automatic storage set to "YES" and automatic resize enabled set to "YES". The following are the automatic resize settings: increase size (bytes) "-1", increase size (percent) "N/A", maximum size (bytes) "-1". The current table space size (bytes) is "6291456". |

Information returned

Table 265. Information returned by the HEALTH_TBS_HI table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table space name |

Table 265. Information returned by the HEALTH_TBS_HI table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| HI_VALUE | SMALLINT | The value of the health indicator. |
| HI_TIMESTAMP | TIMESTAMP | The date and time that the alert was generated. |
| HI_ALERT_STATE | BIGINT | The severity of the alert. |
| HI_ALERT_STATE_DETAIL | VARCHAR(20) | The text description of the HI_ALERT_STATE column. |
| HI_FORMULA | VARCHAR(2048) | The formula used to calculate the health indicator. |
| HI_ADDITIONAL_INFO | VARCHAR(4096) | Additional information about the health indicator. |

HEALTH_TBS_HI_HIS

The HEALTH_TBS_HI_HIS table function returns health indicator history information for table spaces from a health snapshot of a database.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►►—HEALTH_TBS_HI_HIS—(—*dbname*—,—*member*—)—————►◄

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(255) that specifies a valid database name in the same instance as the currently connected database when calling this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify the null value to take the snapshot from the currently connected database.

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT * FROM TABLE(HEALTH_TBS_HI_HIS('','-1')) AS T
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | TABLESPACE_NAME | HI_ID | ... |
|----------------------------|------------------|-------|-----|
| 2006-02-13-12.30.37.181478 | SYSCATSPACE | 2001 | ... |
| 2006-02-13-12.30.37.181478 | SYSCATSPACE | 2001 | ... |
| 2006-02-13-12.30.37.181478 | SYSCATSPACE | 2002 | ... |
| 2006-02-13-12.30.37.181478 | SYSCATSPACE | 2002 | ... |
| 2006-02-13-12.30.37.181478 | SYSCATSPACE | 2003 | ... |
| 2006-02-13-12.30.37.181478 | SYSCATSPACE | 2003 | ... |
| 2006-02-13-12.30.37.181478 | SYSTOOLSPACE | 2001 | ... |
| 2006-02-13-12.30.37.181478 | SYSTOOLSPACE | 2001 | ... |
| 2006-02-13-12.30.37.181478 | SYSTOOLSPACE | 2002 | ... |
| 2006-02-13-12.30.37.181478 | SYSTOOLSPACE | 2002 | ... |
| 2006-02-13-12.30.37.181478 | SYSTOOLSPACE | 2003 | ... |
| 2006-02-13-12.30.37.181478 | SYSTOOLSPACE | 2003 | ... |
| 2006-02-13-12.30.37.181478 | SYSTOOLSTMPSPACE | 2001 | ... |
| 2006-02-13-12.30.37.181478 | SYSTOOLSTMPSPACE | 2001 | ... |
| 2006-02-13-12.30.37.181478 | TEMPSPACE1 | 2001 | ... |
| 2006-02-13-12.30.37.181478 | TEMPSPACE1 | 2001 | ... |
| 2006-02-13-12.30.37.181478 | USERSPACE1 | 2001 | ... |
| 2006-02-13-12.30.37.181478 | USERSPACE1 | 2001 | ... |
| 2006-02-13-12.30.37.181478 | USERSPACE1 | 2002 | ... |
| 2006-02-13-12.30.37.181478 | USERSPACE1 | 2002 | ... |
| 2006-02-13-12.30.37.181478 | USERSPACE1 | 2003 | ... |
| 2006-02-13-12.30.37.181478 | USERSPACE1 | 2003 | ... |

22 record(s) selected.

Output from this query (continued).

| ... | HI_TIMESTAMP | HI_VALUE | HI_ALERT_STATE | HI_ALERT_STATE_DETAIL | ... |
|-----|----------------------------|----------|----------------|-----------------------|-----|
| ... | 2006-02-13-12.16.25.911000 | 0 | 1 | Normal | ... |
| ... | 2006-02-13-12.06.26.168000 | 0 | 1 | Normal | ... |
| ... | 2006-02-13-12.16.25.911000 | 99 | 4 | Alarm | ... |
| ... | 2006-02-13-12.06.26.168000 | 99 | 4 | Alarm | ... |
| ... | 2006-02-13-12.16.25.911000 | 0 | 1 | Normal | ... |
| ... | 2006-02-13-12.06.26.168000 | 0 | 1 | Normal | ... |
| ... | 2006-02-13-12.16.25.911000 | 0 | 1 | Normal | ... |
| ... | 2006-02-13-12.06.26.168000 | 0 | 1 | Normal | ... |
| ... | 2006-02-13-12.16.25.911000 | 62 | 1 | Normal | ... |
| ... | 2006-02-13-12.06.26.168000 | 62 | 1 | Normal | ... |
| ... | 2006-02-13-12.16.25.911000 | 0 | 1 | Normal | ... |
| ... | 2006-02-13-12.06.26.168000 | 0 | 1 | Normal | ... |
| ... | 2006-02-13-12.16.25.911000 | 0 | 1 | Normal | ... |
| ... | 2006-02-13-12.06.26.168000 | 0 | 1 | Normal | ... |

| | | | |
|--------------------------------|-----|----------|-----|
| ... 2006-02-13-12.16.25.911000 | 0 | 1 Normal | ... |
| ... 2006-02-13-12.06.26.168000 | 0 | 1 Normal | ... |
| ... 2006-02-13-12.16.25.911000 | 0 | 1 Normal | ... |
| ... 2006-02-13-12.06.26.168000 | 0 | 1 Normal | ... |
| ... 2006-02-13-12.16.25.911000 | 100 | 4 Alarm | ... |
| ... 2006-02-13-12.06.26.168000 | 100 | 4 Alarm | ... |
| ... 2006-02-13-12.16.25.911000 | 0 | 1 Normal | ... |
| ... 2006-02-13-12.06.26.168000 | 0 | 1 Normal | ... |

Output from this query (continued).

| HI_FORMULA | HI_ADDITIONAL_INFO |
|---------------------------|--|
| ... | ----- |
| ... 0 | - |
| ... 0 | - |
| ... ((9376 / 9468) * 100) | The short term table space growth rate from "02/13/2006 11:16:25.000911" to "02/13/2006 12:16:25.000911" is "N/A" bytes per second and the long term growth rate from "02/12/2006 12:16:25.000911" to "02/13/2006 12:16:25.000911" is "N/A" bytes per second. Time to fullness is projected to be "N/A" and "N/A". The table space is defined with automatic storage set to "YES" and automatic resize enabled set to "YES". |
| ... ((9376 / 9468) * 100) | The short term table space growth rate from "02/13/2006 11:06:26.000168" to "02/13/2006 12:06:26.000168" is "N/A" bytes per second and the long term growth rate from "02/12/2006 12:06:26.000168" to "02/13/2006 12:06:26.000168" is "N/A" bytes per second. Time to fullness is projected to be "N/A" and "N/A". The table space is defined with automatic storage set to "YES" and automatic resize enabled set to "YES". |
| ... 0 | The table space is defined with automatic storage set to "YES" and automatic resize enabled set to "YES". The following are the automatic resize settings: increase size (bytes) "-1", increase size (percent) "N/A", maximum size (bytes) "-1". The current table space size (bytes) is "38797312". |
| ... 0 | The table space is defined with automatic storage set to "YES" and automatic resize enabled set to "YES". The following are the automatic resize settings: increase size (bytes) "-1", increase size (percent) "N/A", maximum size (bytes) "-1". The current table space size (bytes) is "38797312". |
| ... 0 | - |
| ... 0 | - |
| ... ((156 / 252) * 100) | The short term table space growth rate from "02/13/2006 11:16:25.000911" to "02/13/2006 12:16:25.000911" is "N/A" bytes per second and the long term growth rate from "02/12/2006 12:16:25.000911" to "02/13/2006 12:16:25.000911" is "N/A" bytes per second. Time to fullness is projected to be "N/A" and "N/A". The table space is defined with automatic storage set to "YES" and automatic resize enabled set to "YES". |
| ... ((156 / 252) * 100) | The short term table space growth rate from |

```

... 0
... 0
... 0
... 0
... 0
... 0
... 0
... ((1504 / 1504) * 100)
... ((1504 / 1504) * 100)
... 0
... 0

```

"02/13/2006 11:06:26.000168" to
"02/13/2006 12:06:26.000168" is "N/A"
bytes per second and the long term growth
rate from "02/12/2006 12:06:26.000168" to
"02/13/2006 12:06:26.000168" is "N/A" bytes
per second. Time to fullness is projected
to be "N/A" and "N/A". The
table space is defined with automatic
storage set to "YES" and automatic resize
enabled set to "YES".

The table space is defined with automatic
storage set to "YES" and automatic resize
enabled set to "YES". The following are
the automatic resize settings: increase
size (bytes) "-1", increase size (percent)
"N/A", maximum size (bytes) "-1". The
current table space size (bytes) is
"1048576".

The table space is defined with automatic
storage set to "YES" and automatic resize
enabled set to "YES". The following are
the automatic resize settings: increase
size (bytes) "-1", increase size (percent)
"N/A", maximum size (bytes) "-1". The
current table space size (bytes) is
"1048576".

-
-
-
-
-
-

The short term table space growth rate
from "02/13/2006 11:16:25.000911" to
"02/13/2006 12:16:25.000911" is "N/A"
bytes per second and the long term growth
rate from "02/12/2006 12:16:25.000911"
to "02/13/2006 12:16:25.000911" is "N/A"
bytes per second. Time to fullness is
projected to be "N/A" and "N/A".
The table space is defined
with automatic storage set to "YES" and
automatic resize enabled set to "YES".

The short term table space growth rate
from "02/13/2006 11:06:26.000168" to
"02/13/2006 12:06:26.000168" is "N/A"
bytes per second and the long term growth
rate from "02/12/2006 12:06:26.000168"
to "02/13/2006 12:06:26.000168" is "N/A"
bytes per second. Time to fullness is
projected to be "N/A" and "N/A".
The table space is defined
with automatic storage set to "YES" and
automatic resize enabled set to "YES".

The table space is defined with automatic
storage set to "YES" and automatic
resize enabled set to "YES". The
following are the automatic resize
settings: increase size (bytes) "-1",
increase size (percent) "N/A", maximum
size (bytes) "-1". The current table
space size (bytes) is "6291456".

The table space is defined with automatic
storage set to "YES" and automatic
resize enabled set to "YES". The
following are the automatic resize
settings: increase size (bytes) "-1",

increase size (percent) "N/A", maximum size (bytes) "-1". The current table space size (bytes) is "6291456".

Information returned

Table 266. Information returned by the HEALTH_TBS_HI_HIS table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table space name |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| HI_TIMESTAMP | TIMESTAMP | The date and time that the alert was generated. |
| HI_VALUE | SMALLINT | The value of the health indicator. |
| HI_ALERT_STATE | BIGINT | The severity of the alert. |
| HI_ALERT_STATE_DETAIL | VARCHAR(20) | The text description of the HI_ALERT_STATE column. |
| HI_FORMULA | VARCHAR(2048) | The formula used to calculate the health indicator. |
| HI_ADDITIONAL_INFO | VARCHAR(4096) | Additional information about the health indicator. |

HEALTH_TBS_INFO

Returns table space information from a health snapshot of a database.

Important: This table function has been deprecated and might be removed in a future release because the health monitor has been deprecated in Version 9.7. It is not supported in Db2 pureScale environments. For more information, see “Health monitor has been deprecated” at http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.wn.doc/doc/i0055045.html.

Syntax

►► HEALTH_TBS_INFO (—dbname—, —member—) ◀◀

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(255) that specifies a valid database name in the same instance as the currently connected database when calling this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify the null value to take the snapshot from the currently connected database.

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

```
SELECT * FROM TABLE(HEALTH_TBS_INFO('',-1)) AS T
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | TABLESPACE_NAME | ... |
|----------------------------|------------------|-----|
| 2006-02-13-12.30.35.027383 | SYSCATSPACE | ... |
| 2006-02-13-12.30.35.027383 | SYSTOOLSPACE | ... |
| 2006-02-13-12.30.35.027383 | SYSTOOLSTMPSPACE | ... |
| 2006-02-13-12.30.35.027383 | TEMPSPACE1 | ... |
| 2006-02-13-12.30.35.027383 | USERSPACE1 | ... |

5 record(s) selected.

Output from this query (continued).

| ROLLED_UP_ALERT_STATE | ROLLED_UP_ALERT_STATE_DETAIL |
|-----------------------|------------------------------|
| 4 | Alarm |
| 1 | Normal |
| 1 | Normal |
| 1 | Normal |
| 4 | Alarm |

Information returned

Table 267. Information returned by the HEALTH_TBS_INFO table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table space name |
| ROLLED_UP_ALERT_STATE | BIGINT | The most severe alert state captured by this snapshot. |

Table 267. Information returned by the HEALTH_TBS_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|-------------|---|
| ROLLED_UP_ALERT_STATE_DETAIL | VARCHAR(20) | The text description of the ROLLED_UP_ALERT_STATE column. |

LOCKS_HELD administrative view - Retrieve information about locks held

The LOCKS_HELD administrative view returns information about the current locks held.

Note: This administrative view has been deprecated and replaced by the “MON_GET_APPL_LOCKWAIT - Get information about locks for which an application is waiting” on page 561, “MON_GET_LOCKS - List all locks in the currently connected database” on page 674, and “MON_FORMAT_LOCK_NAME - Format the internal lock name and return details” on page 483.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the LOCKS_HELD administrative view
- CONTROL privilege on the LOCKS_HELD administrative view
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Example 1: List the total number of locks held by each table in the database SAMPLE.

```
SELECT TABSCHEMA, TABNAME, COUNT(*) AS NUMBER_OF_LOCKS_HELD
  FROM SYSIBMADM.LOCKS_HELD WHERE DB_NAME = 'SAMPLE'
 GROUP BY DBPARTITIONNUM, TABSCHEMA, TABNAME
```

The following is an example of output for this query.

| TABSCHEMA | TABNAME | NUMBER_OF_LOCKS_HELD |
|-----------|------------|----------------------|
| JESSICAE | EMPLOYEE | 5 |
| JESSICAE | EMP_RESUME | 1 |
| JESSICAE | ORG | 3 |

Example 2: List all the locks that have not escalated in the currently connected database, SAMPLE.

```
SELECT AGENT_ID, TABSCHEMA, TABNAME, LOCK_OBJECT_TYPE, LOCK_MODE,
       LOCK_STATUS FROM SYSIBMADM.LOCKS_HELD WHERE LOCK_ESCALATION = 0
       AND DBPARTITIONNUM = 0
```

The following is an example of output for this query.

| AGENT_ID | TABSCHEMA | TABNAME | LOCK_OBJECT_TYPE | LOCK_MODE | LOCK_STATUS |
|----------|-----------|----------|------------------|-----------|-------------|
| 680 | JESSICAE | EMPLOYEE | INTERNALV_LOCK | S | GRNT |
| 680 | JESSICAE | EMPLOYEE | INTERNALP_LOCK | S | GRNT |

Example 3: List lock information for the locks that are currently held by the application with agent ID 310.

```
SELECT TABSCHEMA, TABNAME, LOCK_OBJECT_TYPE, LOCK_MODE, LOCK_STATUS,
       LOCK_ESCALATION FROM SYSIBMADM.LOCKS_HELD WHERE AGENT_ID = 310
```

The following is an example of output for this query.

| TABSCHEMA | TABNAME | LOCK_OBJECT_TYPE | LOCK_MODE | LOCK_STATUS |
|-----------|------------|------------------|-----------|-------------|
| JESSICAE | EMP_RESUME | TABLE_LOCK | S | GRNT |
| JESSICAE | EMPLOYEE | ROW_LOCK | S | GRNT |

Information returned

Table 268. Information returned by the LOCKS_HELD administrative view

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | Date and time the report was generated. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| APPL_NAME | VARCHAR(256) | appl_name - Application name |
| AUTHID | VARCHAR(128) | auth_id - Authorization ID |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TABNAME | VARCHAR(128) | table_name - Table name |
| TAB_FILE_ID | BIGINT | table_file_id - Table file identification |

Table 268. Information returned by the LOCKS_HELD administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------|-------------|--|
| LOCK_OBJECT_TYPE | VARCHAR(18) | lock_object_type - Lock object type waited on. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • AUTORESIZE_LOCK • AUTOSTORAGE_LOCK • BLOCK_LOCK • EOT_LOCK • INPLACE_REORG_LOCK • INTERNAL_LOCK • INTERNALB_LOCK • INTERNALC_LOCK • INTERNALJ_LOCK • INTERNALL_LOCK • INTERNALO_LOCK • INTERNALQ_LOCK • INTERNALP_LOCK • INTERNALS_LOCK • INTERNALT_LOCK • INTERNALV_LOCK • KEYVALUE_LOCK • ROW_LOCK • SYSBOOT_LOCK • TABLE_LOCK • TABLE_PART_LOCK • TABLESPACE_LOCK • XML_PATH_LOCK • COL_TABLE_SERIALIZE_LOCK |
| LOCK_NAME | VARCHAR(32) | lock_name - Lock name |
| LOCK_MODE | VARCHAR(10) | lock_mode - Lock mode. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |

Table 268. Information returned by the LOCKS_HELD administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------|-------------|--|
| LOCK_STATUS | VARCHAR(10) | lock_status - Lock status. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • CONV • GRNT |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock escalation |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

LOCKWAITS administrative view - Retrieve current lockwaits information

The LOCKWAITS administrative view returns information about database agents working on behalf of applications that are waiting to obtain locks.

Note: This administrative view has been deprecated and replaced by the “MON_LOCKWAITS administrative view - Retrieve metrics for applications that are waiting to obtain locks” on page 441.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the LOCKWAITS administrative view
- CONTROL privilege on the LOCKWAITS administrative view
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Examples

Example 1: List information for all the lock waits for application with agent ID 89.

```
SELECT SUBSTR(TABSCHEMA,1,8) AS TABSCHEMA, SUBSTR(TABNAME,1,15) AS TABNAME,
       LOCK_OBJECT_TYPE, LOCK_MODE, LOCK_MODE_REQUESTED, AGENT_ID_HOLDING_LK
FROM SYSIBMADM.LOCKWAITS WHERE AGENT_ID = 89
```

The following is an example of output for this query.

```
TABSCHEMA TABNAME      LOCK_OBJECT_TYPE LOCK_MODE ...
-----
JESSICAE  T1           ROW_LOCK           X      ...
```

1 record(s) selected.

Output for this query (continued).

```
... LOCK_MODE_REQUESTED AGENT_ID_HOLDING_LK
... -----
... NS                      7
```

Example 2: List the total number of outstanding lock requests per table in the database SAMPLE. By sorting the output by number of requests, tables with the highest contention can be identified.

```
SELECT SUBSTR(TABSCHEMA,1,8) AS TABSCHEMA, SUBSTR(TABNAME, 1, 15)
      AS TABNAME, COUNT(*) AS NUM_OF_LOCK_REQUESTS_WAITING,
      DBPARTITIONNUM
FROM SYSIBMADM.LOCKWAITS WHERE DB_NAME = 'SAMPLE'
GROUP BY TABSCHEMA, TABNAME, DBPARTITIONNUM
ORDER BY NUM_OF_LOCK_REQUESTS_WAITING DESC
```

The following is an example of output for this query.

```
TABSCHEMA TABNAME      NUM_OF_LOCK_REQUESTS_WAITING DBPARTITIONNUM
-----
JESSICAE  T3                      2                0
JESSICAE  T1                      1                0
JESSICAE  T2                      1                0
```

3 record(s) selected.

Information returned

Table 269. Information returned by the LOCKWAITS administrative view

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | Date and time the report was generated. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| APPL_NAME | VARCHAR(256) | appl_name - Application name |
| AUTHID | VARCHAR(128) | auth_id - Authorization ID |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TABNAME | VARCHAR(128) | table_name - Table name |
| SUBSECTION_NUMBER | BIGINT | ss_number - Subsection number |

Table 269. Information returned by the LOCKWAITS administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|-------------|--|
| LOCK_OBJECT_TYPE | VARCHAR(18) | lock_object_type - Lock object type waited on. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • AUTORESIZE_LOCK • AUTOSTORAGE_LOCK • BLOCK_LOCK • EOT_LOCK • INPLACE_REORG_LOCK • INTERNAL_LOCK • INTERNALB_LOCK • INTERNALC_LOCK • INTERNALJ_LOCK • INTERNALL_LOCK • INTERNALO_LOCK • INTERNALQ_LOCK • INTERNALP_LOCK • INTERNALS_LOCK • INTERNALT_LOCK • INTERNALV_LOCK • KEYVALUE_LOCK • ROW_LOCK • SYSBOOT_LOCK • TABLE_LOCK • TABLE_PART_LOCK • TABLESPACE_LOCK • XML_PATH_LOCK • COL_TABLE_SERIALIZE_LOCK |
| LOCK_WAIT_START_TIME | TIMESTAMP | lock_wait_start_time - Lock wait start timestamp |
| LOCK_NAME | VARCHAR(32) | lock_name - Lock name |
| LOCK_MODE | VARCHAR(10) | lock_mode - Lock mode. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |

Table 269. Information returned by the LOCKWAITS administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|--------------|---|
| LOCK_MODE_REQUESTED | VARCHAR(10) | lock_mode_requested - Lock mode requested. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |
| AGENT_ID_HOLDING_LK | BIGINT | agent_id_holding_lock - Agent ID holding lock |
| APPL_ID_HOLDING_LK | VARCHAR(128) | appl_id_holding_lk - Application ID holding lock |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock escalation |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

LOG_UTILIZATION administrative view - Retrieve log utilization information

The LOG_UTILIZATION administrative view returns information about log utilization for the currently connected database. A single row is returned for each database partition.

Important: The LOG_UTILIZATION administrative view is deprecated and has been replaced by the “MON_TRANSACTION_LOG_UTILIZATION administrative view - Retrieve log utilization information” on page 454.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the LOG_UTILIZATION administrative view
- CONTROL privilege on the LOG_UTILIZATION administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

List the log utilization for the currently connected database, SAMPLE.

```
SELECT * FROM SYSIBMADM.LOG_UTILIZATION
```

The following is an example of output for this query.

```
DB_NAME  ... LOG_UTILIZATION_PERCENT TOTAL_LOG_USED_KB  ...
-----  ... -----
SAMPLE   ...                9.75                1989 ...
                                     ...
1 record(s) selected.                ...
```

Output for this query (continued).

```
... TOTAL_LOG_AVAILABLE_KB TOTAL_LOG_USED_TOP_KB DBPARTITIONNUM
... -----
...                18411                1990                0
...
...
```

Usage note

For databases that are configured for infinite logging, the LOG_UTILIZATION_PERCENT and TOTAL_LOG_AVAILABLE_KB will be NULL.

Information returned

Table 270. Information returned by the LOG_UTILIZATION administrative view

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|--|
| DB_NAME | VARCHAR(128) | db_name - Database name |
| LOG_UTILIZATION_PERCENT | DECIMAL(5,2) | Percent utilization of total log space. |
| TOTAL_LOG_USED_KB | BIGINT | total_log_used - Total log space used. This interface returns the value in KB. |
| TOTAL_LOG_AVAILABLE_KB | BIGINT | total_log_available - Total log available. This interface returns the value in KB. |
| TOTAL_LOG_USED_TOP_KB | BIGINT | tot_log_used_top - Maximum total log space used. This interface returns the value in KB. |

Table 270. Information returned by the LOG_UTILIZATION administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------|-----------|--|
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

LONG_RUNNING_SQL administrative view

The LONG_RUNNING_SQL administrative view returns SQL statements executed in the currently connected database. This view can be used to identify long-running SQL statements in the database.

Important: The LONG_RUNNING_SQL administrative view is deprecated and has been replaced by the “MON_CURRENT_SQL - Retrieve key metrics for all activities on all members” on page 435.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the LONG_RUNNING_SQL administrative view
- CONTROL privilege on the LONG_RUNNING_SQL administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMaint
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a report on long running SQL statements in the currently connected database.

```
SELECT SUBSTR(STMT_TEXT, 1, 50) AS STMT_TEXT, AGENT_ID,
       ELAPSED_TIME_MIN, APPL_STATUS, DBPARTITIONNUM
FROM SYSIBMADM.LONG_RUNNING_SQL ORDER BY DBPARTITIONNUM
```

The following is an example of output for this query.

```
STMT_TEXT                                AGENT_ID    ...
-----
select * from dbuser.employee            228 ...
```

```

select * from dbuser.employee          228 ...
select * from dbuser.employee          228 ...
                                         ...
3 record(s) selected.                  ...

```

Output for this query (continued).

```

... ELAPSED_TIME_MIN APPL_STATUS      DBPARTITIONNUM
... -----
...          2 UOWWAIT                  0
...          0 CONNECTED                1
...          0 CONNECTED                2

```

Usage note

This view can be used to identify long-running SQL statements in the database. You can look at the currently running queries to see which statements are the longest running and the current status of the query. Further investigation can be done of the application containing the SQL statement, using agent ID as the unique identifier. If executing a long time and waiting on a lock, you might want to dig deeper using the LOCKWAITS or LOCKS_HELD administrative views. If "waiting on User", this means that the database server is not doing anything but rather is waiting for the application to do something (like issue the next fetch or submit the next SQL statement).

Information returned

Table 271. Information returned by the LONG_RUNNING_SQL administrative view

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | Time the report was generated. |
| ELAPSED_TIME_MIN | INTEGER | Elapsed time of the statement in minutes. |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| APPL_NAME | VARCHAR(256) | appl_name - Application name |

Table 271. Information returned by the `LONG_RUNNING_SQL` administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|--------------|--|
| APPL_STATUS | VARCHAR(22) | <p>appl_status - Application status. This interface returns a text identifier based on the defines in <code>sqlmon.h</code>, and is one of:</p> <ul style="list-style-type: none"> • BACKUP • COMMIT_ACT • COMP • CONNECTED • CONNECTPEND • CREATE_DB • DECOUPLED • DISCONNECTPEND • INTR • IOERROR_WAIT • LOAD • LOCKWAIT • QUIESCE_TABLESPACE • RECOMP • REMOTE_RQST • RESTART • RESTORE • ROLLBACK_ACT • ROLLBACK_TO_SAVEPOINT • TEND • THABRT • THCOMT • TPREP • UNLOAD • UOWEXEC • UOWWAI • WAITFOR_REMOTE |
| AUTHID | VARCHAR(128) | auth_id - Authorization ID |
| INBOUND_COMM_ADDRESS | VARCHAR(32) | inbound_comm_address - Inbound communication address |
| STMT_TEXT | CLOB(16 M) | stmt_text - SQL statement text |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

REG_VARIABLES administrative view - Retrieve database registry settings in use

The `REG_VARIABLES` administrative view returns the database registry settings from all database partitions.

The database registry variable values returned when the REG_VARIABLES administrative view is queried can differ from those returned by the **db2set** command if a database registry variable is configured using the **db2set** command after the instance has been started. The difference occurs because REG_VARIABLES only returns the values that were in effect when the instance was started.

Note: This administrative view has been deprecated and replaced by the ENV_GET_REG_VARIABLES table function.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the REG_VARIABLES administrative view
- CONTROL privilege on the REG_VARIABLES administrative view
- DATAACCESS authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Request the database registry settings that are currently being used.

```
SELECT * from SYSIBMADM.REG_VARIABLES
```

The following is an example of output from this query.

| DBPARTITIONNUM | REG_VAR_NAME | REG_VAR_VALUE | IS_AGGREGATE | AGGREGATE_NAME |
|----------------|-----------------|---------------|--------------|----------------|
| 0 | DB2ADMINSERVER | DB2DAS00 | 0 | - |
| 0 | DB2INSTPROF | D:\SQLLIB | 0 | - |
| 0 | DB2PATH | D:\SQLLIB | 0 | - |
| 0 | DB2SYSTEM | D570 | 0 | - |
| 0 | DB2TEMPDIR | D:\SQLLIB\ | 0 | - |
| 0 | DB2_EXTSECURITY | YES | 0 | - |

6 record(s) selected.

Information returned

Table 272. Information returned by the REG_VARIABLES administrative view

| Column name | Data type | Description |
|----------------|---------------|--|
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| REG_VAR_NAME | VARCHAR(256) | Name of the database registry variable. |
| REG_VAR_VALUE | VARCHAR(2048) | Current setting of the database registry variable. |

Table 272. Information returned by the REG_VARIABLES administrative view (continued)

| Column name | Data type | Description |
|----------------|--------------|--|
| IS_AGGREGATE | SMALLINT | Indicates whether or not the database registry variable is an aggregate variable. The possible return values are 0 if it is not an aggregate variable, and 1 if it is an aggregate variable. |
| AGGREGATE_NAME | VARCHAR(256) | Name of the aggregate if the database registry variable is currently obtaining its value from a configured aggregate. If the registry variable is not being set through an aggregate, or is set through an aggregate but has been overridden, the value of AGGREGATE_NAME is NULL. |
| LEVEL | CHAR(1) | Indicates the level at which the database registry variable acquires its value. The possible return values and the corresponding levels that they represent are: <ul style="list-style-type: none"> • I = instance • G = global • N = database partition • E = environment |

REG_LIST_VARIABLES table function - return registry settings

Note: The REG_LIST_VARIABLES table function has been deprecated and replaced by the ENV_GET_REG_VARIABLES table function.

The REG_LIST_VARIABLES table function returns the Db2 registry settings that are currently being used by the database partition from which the function was invoked.

Syntax

►►—REG_LIST_VARIABLES—(—)—————◄◄

The schema is SYSPROC.

Authorization

EXECUTE privilege on the REG_LIST_VARIABLES table function.

Information returned

Table 273. Information returned by the REG_LIST_VARIABLES table function

| Column name | Data type | Description |
|----------------|---------------|--|
| DBPARTITIONNUM | SMALLINT | The logical partition number of each database partition on which this function runs. |
| REG_VAR_NAME | VARCHAR(256) | The name of the Db2 registry variable. |
| REG_VAR_VALUE | VARCHAR(2048) | The current setting of the Db2 registry variable. |
| IS_AGGREGATE | SMALLINT | Indicates if the Db2 registry variable is an aggregate variable. Possible return values are: <ul style="list-style-type: none">• 0 - Not aggregate• 1 - Aggregate |
| AGGREGATE_NAME | VARCHAR(256) | The name of the aggregate if the Db2 registry variable is currently obtaining its value from a configured aggregate. If the registry variable is not being set through an aggregate or if it is being set by an aggregate that is overridden, the value of AGGREGATE_NAME is NULL. |
| LEVEL | CHAR(1) | Indicates the level at which the Db2 registry variable acquires its value. Possible return value are: <ul style="list-style-type: none">• I - Instance• G - Global• N - Database partition• E - Environment |

Usage notes

The output from the **db2set** command and REG_LIST_VARIABLES might not be identical if a Db2 registry variable is configured using the **db2set** command after the instance is started because REG_LIST_VARIABLES only returns the values that are used when the instance starts.

Example

Request the Db2 registry settings that are currently being used by the current database partition. To format the output into a manageable table, you can use the SUBSTR scalar function to remove trailing blanks from variable length character string column values.

```
SELECT DBPARTITIONNUM,  
       SUBSTR(REG_VAR_NAME, 1, 15) AS REG_VAR_NAME,  
       SUBSTR(REG_VAR_VALUE, 1, 13) AS REG_VAR_VALUE,  
       IS_AGGREGATE,
```

```

SUBSTR(AGGREGATE_NAME, 1, 14) AS AGGREGATE_NAME,
LEVEL
FROM TABLE(SYSPROC.REG_LIST_VARIABLES()) AS REGISTRYINFO

```

This query returns the following output:

| DBPARTITIONNUM | REG_VAR_NAME | REG_VAR_VALUE | IS_AGGREGATE | AGGREGATE_NAME | LEVEL |
|----------------|-----------------|---------------|--------------|----------------|-------|
| 0 | DB2ADMINSERVER | DB2DAS00 | 0 | - | I |
| 0 | DB2INSTPROF | D:\SQLLIB | 0 | - | I |
| 0 | DB2PATH | D:\SQLLIB | 0 | - | I |
| 0 | DB2SYSTEM | D570 | 0 | - | I |
| 0 | DB2TEMPDIR | D:\SQLLIB\ | 0 | - | I |
| 0 | DB2_EXTSECURITY | YES | 0 | - | I |

6 record(s) selected.

SNAPAGENT administrative view and SNAP_GET_AGENT table function - Retrieve agent logical data group application snapshot information

The SNAPAGENT administrative view and the SNAP_GET_AGENT table function return information about agents from an application snapshot, in particular, the agent logical data group.

Important: The SNAPAGENT administrative view and the SNAP_GET_AGENT table function are deprecated and have been replaced by the “MON_GET_AGENT table function - List agents, fenced mode processes, and system entities for the database” on page 551 and “MON_GET_CONNECTION table function - Get connection metrics” on page 588.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPAGENT administrative view”
- “SNAP_GET_AGENT table function” on page 1251

SNAPAGENT administrative view

This administrative view allows you to retrieve agent logical data group application snapshot information for the currently connected database.

Used with the MON_GET_CONNECTION, MON_GET_MEMORY_POOL, and MON_GET_MEMORY_SET table functions and the MON_CURRENT_SQL and SNAPSUBSECTION administrative views, the SNAPAGENT administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS ON database-alias** CLP command, but retrieves data from all database members.

The schema is SYSIBMADM.

Refer to Table 274 on page 1253 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPAGENT administrative view
- CONTROL privilege on the SNAPAGENT administrative view

- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_AGENT table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve all application snapshot information for the currently connected database from the agent logical data group.

```
SELECT * FROM SYSIBMADM.SNAPAGENT
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | DB_NAME | AGENT_ID | ... |
|----------------------------|---------|----------|-----|
| 2005-07-19-11.03.26.740423 | SAMPLE | 101 | ... |
| 2005-07-19-11.03.26.740423 | SAMPLE | 49 | ... |

2 record(s) selected.

Output from this query (continued).

| ... | AGENT_PID | LOCK_TIMEOUT_VAL | DBPARTITIONNUM |
|-----|-----------|------------------|----------------|
| ... | 11980 | -1 | 0 |
| ... | 15940 | -1 | 0 |

SNAP_GET_AGENT table function

The SNAP_GET_AGENT table function returns the same information as the SNAPAGENT administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_AGENT_MEMORY_POOL, SNAP_GET_APPL, SNAP_GET_APPL_INFO, SNAP_GET_STMT and SNAP_GET_SUBSECTION table

functions, the SNAP_GET_AGENT table function provides information equivalent to the **GET SNAPSHOT FOR ALL APPLICATIONS** CLP command, but retrieves data from all database members.

Refer to Table 274 on page 1253 for a complete list of information that can be returned.

Syntax

→ SNAP_GET_AGENT (—*dbname*— [, *member*]) →

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_AGENT table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_AGENT table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve all application snapshot information for all applications in all active databases.

```
SELECT * FROM TABLE(SNAP_GET_AGENT(CAST(NULL AS VARCHAR(128))), -1)) AS T
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | DB_NAME | AGENT_ID | ... |
|----------------------------|---------|----------|-----|
| 2006-01-03-17.21.38.530785 | SAMPLE | 48 | ... |
| 2006-01-03-17.21.38.530785 | SAMPLE | 47 | ... |
| 2006-01-03-17.21.38.530785 | SAMPLE | 46 | ... |
| 2006-01-03-17.21.38.530785 | TESTDB | 30 | ... |
| 2006-01-03-17.21.38.530785 | TESTDB | 29 | ... |
| 2006-01-03-17.21.38.530785 | TESTDB | 28 | ... |

6 record(s) selected.

Output from this query (continued).

| ... | AGENT_PID | LOCK_TIMEOUT_VAL | DBPARTITIONNUM |
|-----|-----------|------------------|----------------|
| ... | 7696 | -1 | 0 |
| ... | 8536 | -1 | 0 |
| ... | 6672 | -1 | 0 |
| ... | 2332 | -1 | 0 |
| ... | 8360 | -1 | 0 |
| ... | 6736 | -1 | 0 |
| ... | | | |

Information returned

Table 274. Information returned by the SNAPAGENT administrative view and the SNAP_GET_AGENT table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| AGENT_PID | BIGINT | agent_pid - Engine dispatchable unit (EDU) |
| LOCK_TIMEOUT_VAL | BIGINT | lock_timeout_val - Lock timeout (seconds) |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function - Retrieve memory_pool logical data group snapshot information

The SNAPAGENT_MEMORY_POOL administrative view and the SNAP_GET_AGENT_MEMORY_POOL table function return information about memory usage at the agent level.

Important: The SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function have been deprecated and replaced by the “MON_GET_MEMORY_POOL - get memory pool information” on page 679 and “MON_GET_MEMORY_SET - get memory set information” on page 682.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPAGENT_MEMORY_POOL administrative view”
- “SNAP_GET_AGENT_MEMORY_POOL table function” on page 1255

SNAPAGENT_MEMORY_POOL administrative view

This administrative view allows you to retrieve the memory_pool logical data group snapshot information about memory usage at the agent level for the currently connected database.

Used with the MON_GET_AGENT and MON_GET_CONNECTION table functions and the MON_CURRENT_SQL and SNAPSUBSECTION administrative views, the SNAPAGENT_MEMORY_POOL administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 275 on page 1257 for a complete list of information that can be returned.

Authorization

One of the following is required to use the view:

- SELECT privilege on the SNAPAGENT_MEMORY_POOL administrative view
- CONTROL privilege on the SNAPAGENT_MEMORY_POOL administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_AGENT_MEMORY_POOL table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of memory pools and their current size.

```
SELECT AGENT_ID, POOL_ID, POOL_CUR_SIZE FROM SYSIBMADM.SNAPAGENT_MEMORY_POOL
```

The following is an example of output from this query.

| AGENT_ID | POOL_ID | POOL_ | CUR_SIZE |
|----------|--------------|-------|----------|
| ----- | ----- | ----- | ----- |
| 48 | APPLICATION | | 65536 |
| 48 | OTHER | | 65536 |
| 48 | APPL_CONTROL | | 65536 |
| 47 | APPLICATION | | 65536 |
| 47 | OTHER | | 131072 |
| 47 | APPL_CONTROL | | 65536 |
| 46 | OTHER | | 327680 |
| 46 | APPLICATION | | 262144 |
| 46 | APPL_CONTROL | | 65536 |

9 record(s) selected.

SNAP_GET_AGENT_MEMORY_POOL table function

The SNAP_GET_AGENT_MEMORY_POOL table function returns the same information as the SNAPAGENT_MEMORY_POOL administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_AGENT, SNAP_GET_APPL, SNAP_GET_APPL_INFO, SNAP_GET_STMT and SNAP_GET_SUBSECTION table functions, the SNAP_GET_AGENT_MEMORY_POOL table function provides information equivalent to the **GET SNAPSHOT FOR ALL APPLICATIONS** CLP command.

Refer to Table 275 on page 1257 for a complete list of information that can be returned.

Syntax

```

>> SNAP_GET_AGENT_MEMORY_POOL ( ( dbname [ , member ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database

name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_AGENT_MEMORY_POOL table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_AGENT_MEMORY_POOL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of memory pools and their current size for all databases.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AGENT_ID, POOL_ID, POOL_CUR_SIZE
FROM TABLE(SNAP_GET_AGENT_MEMORY_POOL(CAST (NULL AS VARCHAR(128)), -1))
AS T
```

The following is an example of output from this query.

| DB_NAME | AGENT_ID | POOL_ID | POOL_CUR_SIZE |
|---------|----------|--------------|---------------|
| SAMPLE | 48 | APPLICATION | 65536 |
| SAMPLE | 48 | OTHER | 65536 |
| SAMPLE | 48 | APPL_CONTROL | 65536 |
| SAMPLE | 47 | APPLICATION | 65536 |
| SAMPLE | 47 | OTHER | 131072 |

| | | | |
|--------|----|--------------|--------|
| SAMPLE | 47 | APPL_CONTROL | 65536 |
| SAMPLE | 46 | OTHER | 327680 |
| SAMPLE | 46 | APPLICATION | 262144 |
| SAMPLE | 46 | APPL_CONTROL | 65536 |
| TESTDB | 30 | APPLICATION | 65536 |
| TESTDB | 30 | OTHER | 65536 |
| TESTDB | 30 | APPL_CONTROL | 65536 |
| TESTDB | 29 | APPLICATION | 65536 |
| TESTDB | 29 | OTHER | 131072 |
| TESTDB | 29 | APPL_CONTROL | 65536 |
| TESTDB | 28 | OTHER | 327680 |
| TESTDB | 28 | APPLICATION | 65536 |
| TESTDB | 28 | APPL_CONTROL | 65536 |

18 record(s) selected.

Information returned

Table 275. Information returned by the SNAPAGENT_MEMORY_POOL administrative view and the SNAP_GET_AGENT_MEMORY_POOL table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| AGENT_PID | BIGINT | agent_pid - Engine dispatchable unit (EDU) |
| POOL_ID | VARCHAR(14) | pool_id - Memory pool identifier. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • APP_GROUP • APPL_CONTROL • APPLICATION • BP • CAT_CACHE • DATABASE • DFM • FCMBP • IMPORT_POOL • LOCK_MGR • MONITOR • OTHER • PACKAGE_CACHE • QUERY • SHARED_SORT • SORT • STATEMENT • STATISTICS • UTILITY |
| POOL_CUR_SIZE | BIGINT | pool_cur_size - Current size of memory pool |

Table 275. Information returned by the SNAPAGENT_MEMORY_POOL administrative view and the SNAP_GET_AGENT_MEMORY_POOL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------|-----------|--|
| POOL_WATERMARK | BIGINT | pool_watermark - Memory pool watermark |
| POOL_CONFIG_SIZE | BIGINT | pool_config_size - Configured size of memory pool |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function - Retrieve appl_info logical data group snapshot information

The SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function return information about applications from an application snapshot, in particular, the appl_info logical data group.

Important: The SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function are deprecated and have been replaced by the “MON_GET_CONNECTION table function - Get connection metrics” on page 588.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPAPPL_INFO administrative view”
- “SNAP_GET_APPL_INFO table function” on page 1259

SNAPAPPL_INFO administrative view

This administrative view allows you to retrieve appl_info logical data group snapshot information for the currently connected database.

Used with the MON_GET_AGENT, MON_GET_CONNECTION, MON_GET_MEMORY_POOL, and MON_GET_MEMORY_SET table functions and the MON_CURRENT_SQL and SNAPSUBSECTION administrative views, the SNAPAPPL_INFO administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS ON database-alias** CLP command, but retrieves data from all database members.

The schema is SYSIBMADM.

Refer to Table 276 on page 1261 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPAPPL_INFO administrative view
- CONTROL privilege on the SNAPAPPL_INFO administrative view

- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_APPL_INFO table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Also, one of the following authorities is required:

- SYSMON
- SYSMAINT
- SYSCTRL
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve the status of the applications connected to the current database.

```
SELECT AGENT_ID, SUBSTR(APPL_NAME,1,10) AS APPL_NAME, APPL_STATUS
FROM SYSIBMADM.SNAPAPPL_INFO
```

The following is an example of output from this query.

| AGENT_ID | APPL_NAME | APPL_STATUS |
|----------|-----------|-------------|
| 101 | db2bp.exe | UOWEXEC |
| 49 | db2bp.exe | CONNECTED |

2 record(s) selected.

SNAP_GET_APPL_INFO table function

The SNAP_GET_APPL_INFO table function returns the same information as the SNAPAPPL_INFO administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_AGENT, SNAP_GET_AGENT_MEMORY_POOL, SNAP_GET_APPL, SNAP_GET_STMT and SNAP_GET_SUBSECTION table functions, the SNAP_GET_APPL_INFO table function provides information equivalent to the **GET SNAPSHOT FOR ALL APPLICATIONS** CLP command, but retrieves data from all database members.

Refer to Table 276 on page 1261 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_APPL_INFO ((*dbname* [, *member*])) ►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_APPL_INFO table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_APPL_INFO table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Retrieve the status of all applications on the connected database member.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AGENT_ID,
       SUBSTR(APPL_NAME,1,10) AS APPL_NAME, APPL_STATUS
FROM TABLE(SNAP_GET_APPL_INFO(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

| DB_NAME | AGENT_ID | APPL_NAME | APPL_STATUS |
|---------|----------|-----------|-------------|
| TOOLSDB | 14 | db2bp.exe | CONNECTED |
| SAMPLE | 15 | db2bp.exe | UOWEXEC |
| SAMPLE | 8 | javaw.exe | CONNECTED |
| SAMPLE | 7 | db2bp.exe | UOWWAIT |

4 record(s) selected.

The following shows what you obtain when you SELECT from the result of the table function.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AUTHORITY_LVL
FROM TABLE(SNAP_GET_APPL_INFO(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

| DB_NAME | AUTHORITY_LVL |
|---------|--|
| TESTDB | SYSADM(GROUP) + DBADM(USER) + CREATETAB(USER, GROUP) +
BINDADD(USER, GROUP) + CONNECT(USER, GROUP) +
CREATE_NOT_FENC(USER) + IMPLICIT_SCHEMA(USER, GROUP) +
LOAD(USER) + CREATE_EXT_RT(USER) + QUIESCE_CONN(USER) |
| TESTDB | SYSADM(GROUP) + DBADM(USER) + CREATETAB(USER, GROUP) +
BINDADD(USER, GROUP) + CONNECT(USER, GROUP) +
CREATE_NOT_FENC(USER) + IMPLICIT_SCHEMA(USER, GROUP) +
LOAD(USER) + CREATE_EXT_RT(USER) + QUIESCE_CONN(USER) |
| TESTDB | SYSADM(GROUP) + DBADM(USER) + CREATETAB(USER, GROUP) +
BINDADD(USER, GROUP) + CONNECT(USER, GROUP) +
CREATE_NOT_FENC(USER) + IMPLICIT_SCHEMA(USER, GROUP) +
LOAD(USER) + CREATE_EXT_RT(USER) + QUIESCE_CONN(USER) |

3 record(s) selected.

Information returned

Table 276. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |

Table 276. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------|-------------|--|
| APPL_STATUS | VARCHAR(22) | <p>appl_status - Application status. This interface returns a text identifier based on the defines in sqlmon.h, and is one of:</p> <ul style="list-style-type: none"> • BACKUP • COMMIT_ACT • COMP • CONNECTED • CONNECTPEND • CREATE_DB • DECOUPLED • DISCONNECTPEND • INTR • IOERROR_WAIT • LOAD • LOCKWAIT • QUIESCE_TABLESPACE • RECOMP • REMOTE_RQST • RESTART • RESTORE • ROLLBACK_ACT • ROLLBACK_TO_SAVEPOINT • TEND • THABRT • THCOMT • TPREP • UNLOAD • UOWEXEC • UOWWAIT • WAITFOR_REMOTE |
| CODEPAGE_ID | BIGINT | codepage_id - ID of code page used by application |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of associated agents |
| COORD_NODE_NUM | SMALLINT | COORD_NODE_NUM is deprecated and is replaced by MEMBER. |

Table 276. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------|--------------|--|
| AUTHORITY_LVL | VARCHAR(512) | <p>authority_bitmap - User authorization level.</p> <p>This interface returns a text identifier based on the database authorities defined in sql.h and their source, and has the following format: authority(source, ...) + authority(source, ...) + ... The source of an authority can be multiple: either from a USER, a GROUP, or a USER and a GROUP.</p> <p>Possible values for "authority":</p> <ul style="list-style-type: none"> • ACCESSCTRL • BINDADD • CONNECT • CREATE_EXT_RT • CREATE_NOT_FENC • CREATETAB • DATAACCESS • DBADM • EXPLAIN • IMPLICIT_SCHEMA • LOAD • LIBADM • QUIESCE_CONN • SECADM • SQLADM • SYSADM • SYSCTRL • SYSMANT • SYSMON • SYSQUIESCE • WLMADM <p>Possible values for "source":</p> <ul style="list-style-type: none"> • USER - authority granted to the user or to a role granted to the user. • GROUP - authority granted to a group to which the user belongs or to a role granted to the group to which the user belongs. |
| CLIENT_PID | BIGINT | client_pid - Client process ID |
| COORD_AGENT_PID | BIGINT | coord_agent_pid - Coordinator agent |

Table 276. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|--|
| STATUS_CHANGE_TIME | TIMESTAMP | status_change_time - Application status change time |
| CLIENT_PLATFORM | VARCHAR(12) | <p>client_platform - Client operating platform. This interface returns a text identifier based on the defines in sqlmon.h,</p> <ul style="list-style-type: none"> • AIX • AIX64 • AS400_DRDA • DOS • DYNIX • HP • HP64 • HPIA • HPIA64 • LINUX • LINUX390 • LINUXIA64 • LINUXPPC • LINUXPPC64 • LINUXX8664 • LINUXZ64 • MAC • MVS_DRDA • NT • NT64 • OS2 • OS390 • SCO • SGI • SNI • SUN • SUN64 • UNKNOWN • UNKNOWN_DRDA • VM_DRDA • VSE_DRDA • WINDOWS |

Table 276. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|---------------|---|
| CLIENT_PROTOCOL | VARCHAR(10) | client_protocol - Client communication protocol. This interface returns a text identifier based on the defines in sqlmon.h, <ul style="list-style-type: none"> • CPIC • LOCAL • NPIPE • TCPIP (for Db2) • TCPIP4 • TCPIP6 |
| TERRITORY_CODE | SMALLINT | territory_code - Database territory code |
| APPL_NAME | VARCHAR(256) | appl_name - Application name |
| APPL_ID | VARCHAR(128) | appl_id - Application ID |
| SEQUENCE_NO | VARCHAR(4) | sequence_no - Sequence number |
| PRIMARY_AUTH_ID | VARCHAR(128) | auth_id - Authorization ID |
| SESSION_AUTH_ID | VARCHAR(128) | session_auth_id - Session authorization ID |
| CLIENT_NNAME | VARCHAR(128) | client_nname - Client name monitor element |
| CLIENT_PRDID | VARCHAR(128) | client_prdid - Client product/version ID |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input database alias |
| CLIENT_DB_ALIAS | VARCHAR(128) | client_db_alias - Database alias used by application |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| DB_PATH | VARCHAR(1024) | db_path - Database path |
| EXECUTION_ID | VARCHAR(128) | execution_id - User login ID |
| CORR_TOKEN | VARCHAR(128) | corr_token - DRDA correlation token |
| TPMON_CLIENT_USERID | VARCHAR(256) | tpmon_client_userid - TP monitor client user ID |
| TPMON_CLIENT_WKSTN | VARCHAR(256) | tpmon_client_wkstn - TP monitor client workstation name |
| TPMON_CLIENT_APP | VARCHAR(256) | tpmon_client_app - TP monitor client application name |
| TPMON_ACC_STR | VARCHAR(255) | tpmon_acc_str - TP monitor client accounting string |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| WORKLOAD_ID | INTEGER | workload_id - Workload ID monitor element |

Table 276. Information returned by the *SNAPAPPL_INFO* administrative view and the *SNAP_GET_APPL_INFO* table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|-----------|--|
| IS_SYSTEM_APPL | SMALLINT | is_system_appl - Is System Application monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| COORD_MEMBER | SMALLINT | coord_member - Coordinator member monitor element |
| COORD_DBPARTITIONNUM | SMALLINT | The coordinating database partition number. |

SNAP_GET_APPL_INFO_V95 table function - Retrieve appl_info logical data group snapshot information

The *SNAP_GET_APPL_INFO_V95* table function returns information about applications from an application snapshot, in particular, the *appl_info* logical data group.

Important: The *SNAP_GET_APPL_INFO_V95* table function is deprecated and has been replaced by the “*MON_GET_CONNECTION* table function - Get connection metrics” on page 588.

Refer to Table 277 on page 1268 for a complete list of information that can be returned.

Syntax

►► *SNAP_GET_APPL_INFO_V95* (—*dbname* — [, *member*]) —►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_APPL_INFO_V95 table function takes a snapshot for the currently connected database and database member number.

Authorization

- One of the following authorizations is required:
- EXECUTE privilege on the SNAP_GET_APPL_INFO_V95 table function
 - DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Retrieve the status of all applications on the connected database member.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AGENT_ID,
       SUBSTR(APPL_NAME,1,10) AS APPL_NAME, APPL_STATUS
FROM TABLE(SNAP_GET_APPL_INFO_V95(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

| DB_NAME | AGENT_ID | APPL_NAME | APPL_STATUS |
|---------|----------|-----------|-------------|
| TOOLSDB | 14 | db2bp.exe | CONNECTED |
| SAMPLE | 15 | db2bp.exe | UOWEXEC |
| SAMPLE | 8 | javaw.exe | CONNECTED |
| SAMPLE | 7 | db2bp.exe | UOWWAIT |

4 record(s) selected.

The following shows what you obtain when you SELECT from the result of the table function.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AUTHORITY_LVL
FROM TABLE(SNAP_GET_APPL_INFO_V95(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

| DB_NAME | AUTHORITY_LVL |
|---------|--|
| TESTDB | SYSADM(GROUP) + DBADM(USER) + CREATETAB(USER, GROUP) +
BINDADD(USER, GROUP) + CONNECT(USER, GROUP) +
CREATE_NOT_FENC(USER) + IMPLICIT_SCHEMA(USER, GROUP) +
LOAD(USER) + CREATE_EXT_RT(USER) + QUIESCE_CONN(USER) |
| TESTDB | SYSADM(GROUP) + DBADM(USER) + CREATETAB(USER, GROUP) +
BINDADD(USER, GROUP) + CONNECT(USER, GROUP) +
CREATE_NOT_FENC(USER) + IMPLICIT_SCHEMA(USER, GROUP) + |

```

LOAD(USER) + CREATE_EXT_RT(USER) + QUIESCE_CONN(USER)
TESTDB  SYSADM(GROUP) + DBADM(USER) + CREATETAB(USER, GROUP) +
        BINDADD(USER, GROUP) + CONNECT(USER, GROUP) +
        CREATE_NOT_FENC(USER) + IMPLICIT_SCHEMA(USER, GROUP) +
        LOAD(USER) + CREATE_EXT_RT(USER) + QUIESCE_CONN(USER)

```

3 record(s) selected.

Information returned

Table 277. Information returned by the SNAP_GET_APPL_INFO_V95 table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| APPL_STATUS | VARCHAR(22) | <p>appl_status - Application status. This interface returns a text identifier based on the defines in sqlmon.h, and is one of:</p> <ul style="list-style-type: none"> • BACKUP • COMMIT_ACT • COMP • CONNECTED • CONNECTPEND • CREATE_DB • DECOUPLED • DISCONNECTPEND • INTR • IOERROR_WAIT • LOAD • LOCKWAIT • QUIESCE_TABLESPACE • RECOMP • REMOTE_RQST • RESTART • RESTORE • ROLLBACK_ACT • ROLLBACK_TO_SAVEPOINT • TEND • THABRT • THCOMT • TPREP • UNLOAD • UOWEXEC • UOWWAIT • WAITFOR_REMOTE |
| CODEPAGE_ID | BIGINT | codepage_id - ID of code page used by application |

Table 277. Information returned by the `SNAP_GET_APPL_INFO_V95` table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------|--------------|--|
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of associated agents |
| COORD_NODE_NUM | SMALLINT | coord_node - Coordinating node |
| AUTHORITY_LVL | VARCHAR(512) | <p>authority_bitmap - User authorization level.</p> <p>This interface returns a text identifier based on the database authorities defined in <code>sql.h</code> and their source, and has the following format: <i>authority(source, ...)</i> + <i>authority(source, ...)</i> + ... The source of an authority can be multiple: either from a USER, a GROUP, or a USER and a GROUP.</p> <p>Possible values for "authority":</p> <ul style="list-style-type: none"> • ACCESSCTRL • BINDADD • CONNECT • CREATE_EXT_RT • CREATE_NOT_FENC • CREATETAB • DATAACCESS • DBADM • EXPLAIN • IMPLICIT_SCHEMA • LOAD • LIBADM • QUIESCE_CONN • SECADM • SQLADM • SYSADM • SYSCTRL • SYSMAINT • SYSMON • SYSQUIESCE • WLMADM <p>Possible values for "source":</p> <ul style="list-style-type: none"> • USER - authority granted to the user or to a role granted to the user. • GROUP - authority granted to a group to which the user belongs or to a role granted to the group to which the user belongs. |
| CLIENT_PID | BIGINT | client_pid - Client process ID |

Table 277. Information returned by the SNAP_GET_APPL_INFO_V95 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|--|
| COORD_AGENT_PID | BIGINT | coord_agent_pid - Coordinator agent |
| STATUS_CHANGE_TIME | TIMESTAMP | status_change_time - Application status change time |
| CLIENT_PLATFORM | VARCHAR(12) | <p>client_platform - Client operating platform. This interface returns a text identifier based on the defines in sqlmon.h,</p> <ul style="list-style-type: none"> • AIX • AIX64 • AS400_DRDA • DOS • DYNIX • HP • HP64 • HPIA • HPIA64 • LINUX • LINUX390 • LINUXIA64 • LINUXPPC • LINUXPPC64 • LINUXX8664 • LINUXZ64 • MAC • MVS_DRDA • NT • NT64 • OS2 • OS390 • SCO • SGI • SNI • SUN • SUN64 • UNKNOWN • UNKNOWN_DRDA • VM_DRDA • VSE_DRDA • WINDOWS |

Table 277. Information returned by the SNAP_GET_APPL_INFO_V95 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|---------------|---|
| CLIENT_PROTOCOL | VARCHAR(10) | client_protocol - Client communication protocol. This interface returns a text identifier based on the defines in sqlmon.h, <ul style="list-style-type: none"> • CPIC • LOCAL • NPIPE • TCPIP (for Db2) • TCPIP4 • TCPIP6 |
| TERRITORY_CODE | SMALLINT | territory_code - Database territory code |
| APPL_NAME | VARCHAR(256) | appl_name - Application name |
| APPL_ID | VARCHAR(128) | appl_id - Application ID |
| SEQUENCE_NO | VARCHAR(4) | sequence_no - Sequence number |
| PRIMARY_AUTH_ID | VARCHAR(128) | auth_id - Authorization ID |
| SESSION_AUTH_ID | VARCHAR(128) | session_auth_id - Session authorization ID |
| CLIENT_NNAME | VARCHAR(128) | client_nname - Client name monitor element |
| CLIENT_PRDID | VARCHAR(128) | client_prdid - Client product/version ID |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input database alias |
| CLIENT_DB_ALIAS | VARCHAR(128) | client_db_alias - Database alias used by application |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| DB_PATH | VARCHAR(1024) | db_path - Database path |
| EXECUTION_ID | VARCHAR(128) | execution_id - User login ID |
| CORR_TOKEN | VARCHAR(128) | corr_token - DRDA correlation token |
| TPMON_CLIENT_USERID | VARCHAR(256) | tpmon_client_userid - TP monitor client user ID |
| TPMON_CLIENT_WKSTN | VARCHAR(256) | tpmon_client_wkstn - TP monitor client workstation name |
| TPMON_CLIENT_APP | VARCHAR(256) | tpmon_client_app - TP monitor client application name |
| TPMON_ACC_STR | VARCHAR(255) | tpmon_acc_str - TP monitor client accounting string |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| WORKLOAD_ID | INTEGER | workload_id - Workload ID monitor element |

Table 277. Information returned by the SNAP_GET_APPL_INFO_V95 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------|-----------|--|
| IS_SYSTEM_APPL | SMALLINT | is_system_appl - Is System Application monitor element |

SNAPAPPL administrative view and SNAP_GET_APPL table function - Retrieve appl logical data group snapshot information

The SNAPAPPL administrative view and the SNAP_GET_APPL table function return information about applications from an application snapshot, in particular, the appl logical data group.

Important: The SNAPAPPL administrative view and the SNAP_GET_APPL table function are deprecated and have been replaced by the “MON_GET_CONNECTION table function - Get connection metrics” on page 588.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPAPPL administrative view”
- “SNAP_GET_APPL table function” on page 1273

SNAPAPPL administrative view

This administrative view allows you to retrieve appl logical data group snapshot information for the currently connected database.

Used with the MON_GET_AGENT, MON_GET_CONNECTION, MON_GET_MEMORY_POOL, and MON_GET_MEMORY_SET table functions and the MON_CURRENT_SQL and SNAPSUBSECTION administrative views, the SNAPAPPL administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS ON database-alias** CLP command, but retrieves data from all database members.

The schema is SYSIBMADM.

Refer to Table 278 on page 1275 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPAPPL administrative view
- CONTROL privilege on the SNAPAPPL administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following privileges or authorities is required to use the table function:

- EXECUTE privilege on the SNAP_GET_APPL table function

- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve details on rows read and written for each application in the connected database.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AGENT_ID, ROWS_READ, ROWS_WRITTEN
FROM SYSIBMADM.SNAPAPPL
```

The following is an example of output from this query.

| DB_NAME | AGENT_ID | ROWS_READ | ROWS_WRITTEN |
|---------|----------|-----------|--------------|
| SAMPLE | | 7 | 25 |

1 record(s) selected.

SNAP_GET_APPL table function

The SNAP_GET_APPL table function returns the same information as the SNAPAPPL administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_AGENT, SNAP_GET_AGENT_MEMORY_POOL, SNAP_GET_APPL_INFO, SNAP_GET_STMT and SNAP_GET_SUBSECTION table functions, the SNAP_GET_APPL table function provides information equivalent to the **GET SNAPSHOT FOR ALL APPLICATIONS** CLP command, but retrieves data from all database members.

Refer to Table 278 on page 1275 for a complete list of information that can be returned.

Syntax

```
➤➤ SNAP_GET_APPL ( ( dbname [ , member ] ) ) ➤➤
```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_APPL table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_APPL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve details on rows read and written for each application for all active databases.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AGENT_ID, ROWS_READ, ROWS_WRITTEN
FROM TABLE (SNAP_GET_APPL(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

| DB_NAME | AGENT_ID | ROWS_READ | ROWS_WRITTEN |
|---------|----------|-----------|--------------|
| WSDB | 679 | 0 | 0 |

| | | | |
|------|-----|----|---|
| WSDB | 461 | 3 | 0 |
| WSDB | 460 | 4 | 0 |
| TEST | 680 | 4 | 0 |
| TEST | 455 | 6 | 0 |
| TEST | 454 | 0 | 0 |
| TEST | 453 | 50 | 0 |

Information returned

Table 278. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| UOW_LOG_SPACE_USED | BIGINT | uow_log_space_used - Unit of work log space used |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| INACT_STMTHIST_SZ | BIGINT | inact_stmthist_sz - Statement history list size |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |

Table 278. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|-----------|---|
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| LOCKS_HELD | BIGINT | locks_held - Locks held |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive lock escalations |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |
| TOTAL_SORTS | BIGINT | total_sorts - Total sorts |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total sort time |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| COMMIT_SQL_STMTS | BIGINT | commit_sql_stmts - Commit statements attempted |
| ROLLBACK_SQL_STMTS | BIGINT | rollback_sql_stmts - Rollback statements attempted |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL statements attempted |
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL statements attempted |

Table 278. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|-----------|---|
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed statement operations |
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL statements executed |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data definition language (DDL) SQL statements |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - UPDATE/INSERT/DELETE SQL statements executed |
| INT_AUTO_REBINDS | BIGINT | int_auto_rebinds - Internal automatic rebinds |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal rows deleted |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal rows updated |
| INT_COMMITS | BIGINT | int_commits - Internal commits |
| INT_ROLLBACKS | BIGINT | int_rollback - Internal rollbacks |
| INT_DEADLOCK_ROLLBACKS | BIGINT | int_deadlock_rollback - Internal rollbacks due to deadlock |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated |
| ROWS_SELECTED | BIGINT | rows_selected - Rows selected |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/precompiles attempted |
| OPEN_REM_CURS | BIGINT | open_rem_curs - Open remote cursors |
| OPEN_REM_CURS_BLK | BIGINT | open_rem_curs_blk - Open remote cursors with blocking |
| REJ_CURS_BLK | BIGINT | rej_curs_blk - Rejected block cursor requests |
| ACC_CURS_BLK | BIGINT | acc_curs_blk - Accepted block cursor requests |
| SQL_REQS_SINCE_COMMIT | BIGINT | sql_reqs_since_commit - SQL requests since last commit |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal rows inserted |
| OPEN_LOC_CURS | BIGINT | open_loc_curs - Open local cursors |
| OPEN_LOC_CURS_BLK | BIGINT | open_loc_curs_blk - Open local cursors with blocking |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups |

Table 278. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|---|
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog cache overflows |
| NUM_AGENTS | BIGINT | num_agents - Number of agents working on a statement |
| AGENTS_STOLEN | BIGINT | agents_stolen - Stolen agents |
| ASSOCIATED_AGENTS_TOP | BIGINT | associated_agents_top - Maximum number of associated agents |
| APPL_PRIORITY | BIGINT | appl_priority - Application agent priority |
| APPL_PRIORITY_TYPE | VARCHAR(16) | appl_priority_type - Application priority type. This interface returns a text identifier, based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • DYNAMIC_PRIORITY • FIXED_PRIORITY |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - Section lookups |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Section inserts |
| LOCKS_WAITING | BIGINT | locks_waiting - Current agents waiting on locks |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total hash joins |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total hash loops |
| HASH_JOIN_OVERFLOWS | BIGINT | hash_join_overflows - Hash join overflows |
| HASH_JOIN_SMALL_OVERFLOWS | BIGINT | hash_join_small_overflows - Hash join small overflows |
| APPL_IDLE_TIME | BIGINT | appl_idle_time - Application idle time |
| UOW_LOCK_WAIT_TIME | BIGINT | uow_lock_wait_time - Total time unit of work waited on locks |

Table 278. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|-------------|--|
| UOW_COMP_STATUS | VARCHAR(14) | uow_comp_status - Unit of work completion status. This interface returns a text identifier, based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • APPL_END • UOWABEND • UOWCOMMIT • UOWDEADLOCK • UOWLOCKTIMEOUT • UOWROLLBACK • UOWUNKNOWN |
| AGENT_USR_CPU_TIME_S | BIGINT | agent_usr_cpu_time - User CPU time used by agent (in seconds)* |
| AGENT_USR_CPU_TIME_MS | BIGINT | agent_usr_cpu_time - User CPU time used by agent (fractional, in microseconds)* |
| AGENT_SYS_CPU_TIME_S | BIGINT | agent_sys_cpu_time - System CPU time used by agent (in seconds)* |
| AGENT_SYS_CPU_TIME_MS | BIGINT | agent_sys_cpu_time - System CPU time used by agent (fractional, in microseconds)* |
| APPL_CON_TIME | TIMESTAMP | appl_con_time - Connection request start timestamp |
| CONN_COMPLETE_TIME | TIMESTAMP | conn_complete_time - Connection request completion timestamp |
| LAST_RESET | TIMESTAMP | last_reset - Last reset timestamp |
| UOW_START_TIME | TIMESTAMP | uow_start_time - Unit of work start timestamp |
| UOW_STOP_TIME | TIMESTAMP | uow_stop_time - Unit of work stop timestamp |
| PREV_UOW_STOP_TIME | TIMESTAMP | prev_uow_stop_time - Previous unit of work completion timestamp |
| UOW_ELAPSED_TIME_S | BIGINT | uow_elapsed_time - Most recent unit of work elapsed time (in seconds)* |
| UOW_ELAPSED_TIME_MS | BIGINT | uow_elapsed_time - Most recent unit of work elapsed time (fractional, in microseconds)* |
| ELAPSED_EXEC_TIME_S | BIGINT | elapsed_exec_time - Statement execution elapsed time (in seconds)* |
| ELAPSED_EXEC_TIME_MS | BIGINT | elapsed_exec_time - Statement execution elapsed time (fractional, in microseconds)* |
| INBOUND_COMM_ADDRESS | VARCHAR(32) | inbound_comm_address - Inbound communication address |

Table 278. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--|-----------|--|
| LOCK_TIMEOUT_VAL | BIGINT | lock_timeout_val - Lock timeout (seconds) |
| PRIV_WORKSPACE_NUM_OVERFLOWS | BIGINT | priv_workspace_num_overflows - Private workspace overflows |
| PRIV_WORKSPACE_SECTION_INSERTS | BIGINT | priv_workspace_section_inserts - Private workspace section inserts |
| PRIV_WORKSPACE_SECTION_LOOKUPS | BIGINT | priv_workspace_section_lookups - Private workspace section lookups |
| PRIV_WORKSPACE_SIZE_TOP | BIGINT | priv_workspace_size_top - Maximum private workspace size |
| SHR_WORKSPACE_NUM_OVERFLOWS | BIGINT | shr_workspace_num_overflows - Shared workspace overflows |
| SHR_WORKSPACE_SECTION_INSERTS | BIGINT | shr_workspace_section_inserts - Shared workspace section inserts |
| SHR_WORKSPACE_SECTION_LOOKUPS | BIGINT | shr_workspace_section_lookups - Shared workspace section lookups |
| SHR_WORKSPACE_SIZE_TOP | BIGINT | shr_workspace_size_top - Maximum shared workspace size |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| CAT_CACHE_SIZE_TOP | BIGINT | cat_cache_size_top - Catalog cache high water mark |
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP functions |
| OLAP_FUNC_OVERFLOWS | BIGINT | olap_func_overflows - OLAP function overflows |
| MEMBER | SMALLINT | member - Database member monitor element |
| <p>* To calculate the total time spent for the monitor element that this column is based on, you must add the full seconds reported in the column for this monitor element that ends with _S to the fractional seconds reported in the column for this monitor element that ends with _MS, using the following formula: $(\text{monitor-element-name_S} \times 1,000,000 + \text{monitor-element-name_MS}) \div 1,000,000$. For example, $(\text{ELAPSED_EXEC_TIME_S} \times 1,000,000 + \text{ELAPSED_EXEC_TIME_MS}) \div 1,000,000$.</p> | | |

SNAP_GET_APPL_V95 table function - Retrieve appl logical data group snapshot information

The SNAP_GET_APPL_V95 table function returns information about applications from an application snapshot, in particular, the appl logical data group.

Important: The SNAP_GET_APPL_V95 table function is deprecated and has been replaced by the “MON_GET_CONNECTION table function - Get connection metrics” on page 588.

Refer to Table 279 on page 1282 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_APPL_V95 (—*dbname* — [, *member*]) —►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_APPL_V95 table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_APPL_V95 table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve details on rows read and written for each application for all active databases.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AGENT_ID, ROWS_READ, ROWS_WRITTEN
FROM TABLE (SNAP_GET_APPL_V95(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

| DB_NAME | AGENT_ID | ROWS_READ | ROWS_WRITTEN |
|---------|----------|-----------|--------------|
| WSDB | 679 | 0 | 0 |
| WSDB | 461 | 3 | 0 |
| WSDB | 460 | 4 | 0 |
| TEST | 680 | 4 | 0 |
| TEST | 455 | 6 | 0 |
| TEST | 454 | 0 | 0 |
| TEST | 453 | 50 | 0 |

Information returned

Table 279. Information returned by the SNAP_GET_APPL_V95 table function

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| UOW_LOG_SPACE_USED | BIGINT | uow_log_space_used - Unit of work log space used |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| STMT_HISTORY_LIST_SIZE | BIGINT | stmt_history_list_size - Statement history list size |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |

Table 279. Information returned by the SNAP_GET_APPL_V95 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| LOCKS_HELD | BIGINT | locks_held - Locks held |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive lock escalations |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |
| TOTAL_SORTS | BIGINT | total_sorts - Total sorts |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total sort time |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| COMMIT_SQL_STMTS | BIGINT | commit_sql_stmts - Commit statements attempted |

Table 279. Information returned by the SNAP_GET_APPL_V95 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|-----------|---|
| ROLLBACK_SQL_STMTS | BIGINT | rollback_sql_stmts - Rollback statements attempted |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL statements attempted |
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL statements attempted |
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed statement operations |
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL statements executed |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data definition language (DDL) SQL statements |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - UPDATE/INSERT/DELETE SQL statements executed |
| INT_AUTO_REBINDS | BIGINT | int_auto_rebinds - Internal automatic rebinds |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal rows deleted |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal rows updated |
| INT_COMMITS | BIGINT | int_commits - Internal commits |
| INT_ROLLBACKS | BIGINT | int_rollback - Internal rollbacks |
| INT_DEADLOCK_ROLLBACKS | BIGINT | int_deadlock_rollback - Internal rollbacks due to deadlock |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated |
| ROWS_SELECTED | BIGINT | rows_selected - Rows selected |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/precompiles attempted |
| OPEN_REM_CURS | BIGINT | open_rem_curs - Open remote cursors |
| OPEN_REM_CURS_BLK | BIGINT | open_rem_curs_blk - Open remote cursors with blocking |
| REJ_CURS_BLK | BIGINT | rej_curs_blk - Rejected block cursor requests |
| ACC_CURS_BLK | BIGINT | acc_curs_blk - Accepted block cursor requests |
| SQL_REQS_SINCE_COMMIT | BIGINT | sql_reqs_since_commit - SQL requests since last commit |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal rows inserted |

Table 279. Information returned by the SNAP_GET_APPL_V95 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|---|
| OPEN_LOC_CURS | BIGINT | open_loc_curs - Open local cursors |
| OPEN_LOC_CURS_BLK | BIGINT | open_loc_curs_blk - Open local cursors with blocking |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog cache overflows |
| NUM_AGENTS | BIGINT | num_agents - Number of agents working on a statement |
| AGENTS_STOLEN | BIGINT | agents_stolen - Stolen agents |
| ASSOCIATED_AGENTS_TOP | BIGINT | associated_agents_top - Maximum number of associated agents |
| APPL_PRIORITY | BIGINT | appl_priority - Application agent priority |
| APPL_PRIORITY_TYPE | VARCHAR(16) | appl_priority_type - Application priority type. This interface returns a text identifier, based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • DYNAMIC_PRIORITY • FIXED_PRIORITY |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - Section lookups |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Section inserts |
| LOCKS_WAITING | BIGINT | locks_waiting - Current agents waiting on locks |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total hash joins |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total hash loops |
| HASH_JOIN_OVERFLOWS | BIGINT | hash_join_overflows - Hash join overflows |
| HASH_JOIN_SMALL_OVERFLOWS | BIGINT | hash_join_small_overflows - Hash join small overflows |
| APPL_IDLE_TIME | BIGINT | appl_idle_time - Application idle time |
| UOW_LOCK_WAIT_TIME | BIGINT | uow_lock_wait_time - Total time unit of work waited on locks |

Table 279. Information returned by the SNAP_GET_APPL_V95 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|-------------|--|
| UOW_COMP_STATUS | VARCHAR(14) | uow_comp_status - Unit of work completion status. This interface returns a text identifier, based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • APPL_END • UOWABEND • UOWCOMMIT • UOWDEADLOCK • UOWLOCKTIMEOUT • UOWROLLBACK • UOWUNKNOWN |
| AGENT_USR_CPU_TIME_S | BIGINT | agent_usr_cpu_time - User CPU time used by agent (in seconds)* |
| AGENT_USR_CPU_TIME_MS | BIGINT | agent_usr_cpu_time - User CPU time used by agent (fractional, in microseconds)* |
| AGENT_SYS_CPU_TIME_S | BIGINT | agent_sys_cpu_time - System CPU time used by agent (in seconds)* |
| AGENT_SYS_CPU_TIME_MS | BIGINT | agent_sys_cpu_time - System CPU time used by agent (fractional, in microseconds)* |
| APPL_CON_TIME | TIMESTAMP | appl_con_time - Connection request start timestamp |
| CONN_COMPLETE_TIME | TIMESTAMP | conn_complete_time - Connection request completion timestamp |
| LAST_RESET | TIMESTAMP | last_reset - Last reset timestamp |
| UOW_START_TIME | TIMESTAMP | uow_start_time - Unit of work start timestamp |
| UOW_STOP_TIME | TIMESTAMP | uow_stop_time - Unit of work stop timestamp |
| PREV_UOW_STOP_TIME | TIMESTAMP | prev_uow_stop_time - Previous unit of work completion timestamp |
| UOW_ELAPSED_TIME_S | BIGINT | uow_elapsed_time - Most recent unit of work elapsed time (in seconds)* |
| UOW_ELAPSED_TIME_MS | BIGINT | uow_elapsed_time - Most recent unit of work elapsed time (fractional, in microseconds)* |
| ELAPSED_EXEC_TIME_S | BIGINT | elapsed_exec_time - Statement execution elapsed time (in seconds)* |
| ELAPSED_EXEC_TIME_MS | BIGINT | elapsed_exec_time - Statement execution elapsed time (fractional, in microseconds)* |
| INBOUND_COMM_ADDRESS | VARCHAR(32) | inbound_comm_address - Inbound communication address |

Table 279. Information returned by the SNAP_GET_APPL_V95 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--|-----------|--|
| LOCK_TIMEOUT_VAL | BIGINT | lock_timeout_val - Lock timeout (seconds) |
| PRIV_WORKSPACE_NUM_OVERFLOWS | BIGINT | priv_workspace_num_overflows - Private workspace overflows |
| PRIV_WORKSPACE_SECTION_INSERTS | BIGINT | priv_workspace_section_inserts - Private workspace section inserts |
| PRIV_WORKSPACE_SECTION_LOOKUPS | BIGINT | priv_workspace_section_lookups - Private workspace section lookups |
| PRIV_WORKSPACE_SIZE_TOP | BIGINT | priv_workspace_size_top - Maximum private workspace size |
| SHR_WORKSPACE_NUM_OVERFLOWS | BIGINT | shr_workspace_num_overflows - Shared workspace overflows |
| SHR_WORKSPACE_SECTION_INSERTS | BIGINT | shr_workspace_section_inserts - Shared workspace section inserts |
| SHR_WORKSPACE_SECTION_LOOKUPS | BIGINT | shr_workspace_section_lookups - Shared workspace section lookups |
| SHR_WORKSPACE_SIZE_TOP | BIGINT | shr_workspace_size_top - Maximum shared workspace size |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| CAT_CACHE_SIZE_TOP | BIGINT | cat_cache_size_top - Catalog cache high water mark |
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP Functions monitor element |
| OLAP_FUNC_OVERFLOWS | BIGINT | olap_func_overflows - OLAP Function Overflows monitor element |
| <p>* To calculate the total time spent for the monitor element that this column is based on, you must add the full seconds reported in the column for this monitor element that ends with _S to the fractional seconds reported in the column for this monitor element that ends with _MS, using the following formula: $(\text{monitor-element-name_S} \times 1,000,000 + \text{monitor-element-name_MS}) \div 1,000,000$. For example, $(\text{ELAPSED_EXEC_TIME_S} \times 1,000,000 + \text{ELAPSED_EXEC_TIME_MS}) \div 1,000,000$.</p> | | |

SNAPBP administrative view and SNAP_GET_BP table function - Retrieve bufferpool logical group snapshot information

The SNAPBP administrative view and the SNAP_GET_BP table function return information about buffer pools from a bufferpool snapshot, in particular, the bufferpool logical data group.

Important: The SNAPBP administrative view and the SNAP_GET_BP table function are deprecated and have been replaced by the “MON_GET_BUFFERPOOL table function - Get buffer pool metrics” on page 571.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPBP administrative view”
- “SNAP_GET_BP table function” on page 1289

SNAPBP administrative view

This administrative view allows you to retrieve bufferpool logical group snapshot information for the currently connected database.

Used with the MON_GET_BUFFERPOOL table function, the SNAPBP administrative view provides the data equivalent to the **GET SNAPSHOT FOR BUFFERPOOLS ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 280 on page 1290 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPBP administrative view
- CONTROL privilege on the SNAPBP administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_BP table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve data and index writes for all the bufferpools of the currently connected database.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME,SUBSTR(BP_NAME,1,15)
      AS BP_NAME,POOL_DATA_WRITES,POOL_INDEX_WRITES
FROM SYSIBMADM.SNAPBP
```

The following is an example of output from this query.

| DB_NAME | BP_NAME | POOL_DATA_WRITES | POOL_INDEX_WRITES |
|---------|----------------|------------------|-------------------|
| TEST | IBMDEFAULTBP | 0 | 0 |
| TEST | IBMSYSTEMBP4K | 0 | 0 |
| TEST | IBMSYSTEMBP8K | 0 | 0 |
| TEST | IBMSYSTEMBP16K | 0 | 0 |
| TEST | IBMSYSTEMBP32K | 0 | 0 |

5 record(s) selected

SNAP_GET_BP table function

The SNAP_GET_BP table function returns the same information as the SNAPBP administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_BP_PART table function, the SNAP_GET_BP table function provides the data equivalent to the **GET SNAPSHOT FOR ALL BUFFERPOOLS** CLP command.

Refer to Table 280 on page 1290 for a complete list of information that can be returned.

Syntax

```

-- SNAP_GET_BP ( ( dbname [ , member ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_BP table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_BP table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve total physical and logical reads for all bufferpools for all active databases for the currently connected database member.

```
SELECT SUBSTR(T.DB_NAME,1,10) AS DB_NAME,  
       SUBSTR(T.BP_NAME,1,20) AS BP_NAME,  
       (T.POOL_DATA_L_READS+T.POOL_INDEX_L_READS) AS TOTAL_LOGICAL_READS,  
       (T.POOL_DATA_P_READS+T.POOL_INDEX_P_READS) AS TOTAL_PHYSICAL_READS,  
       T.DBPARTITIONNUM  
FROM TABLE(SNAP_GET_BP(CAST(NULL AS VARCHAR(128)), -1)) AS T
```

The following is an example of output from this query.

| DB_NAME | BP_NAME | TOTAL_LOGICAL_READS | ... |
|---------|--------------|---------------------|-----|
| ----- | ----- | ----- | ... |
| SAMPLE | IBMDEFAULTBP | 0 | ... |
| TOOLSDB | IBMDEFAULTBP | 0 | ... |
| TOOLSDB | BP32K0000 | 0 | ... |

3 record(s) selected.

Output from this query (continued).

| ... | TOTAL_PHYSICAL_READS | DBPARTITIONNUM |
|-----|----------------------|----------------|
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |

Information returned

Table 280. Information returned by the SNAPBP administrative view and the SNAP_GET_BP table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| BP_NAME | VARCHAR(128) | bp_name - Buffer pool name |
| DB_NAME | VARCHAR(128) | db_name - Database name |

Table 280. Information returned by the SNAPBP administrative view and the SNAP_GET_BP table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|---------------|--|
| DB_PATH | VARCHAR(1024) | db_path - Database path |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input database alias |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes |
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer pool asynchronous read time |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests |

Table 280. Information returned by the SNAPBP administrative view and the SNAP_GET_BP table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|---|
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| FILES_CLOSED | BIGINT | files_closed - Database files closed |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer pool no victim buffers |
| PAGES_FROM_BLOCK_IOS | BIGINT | pages_from_block_ios - Total number of pages read by block I/O |
| PAGES_FROM_VECTORED_IOS | BIGINT | pages_from_vectored_ios - Total pages read by vectored I/O |
| VECTORED_IOS | BIGINT | vectored_ios - Number of vectored I/O requests |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

Table 280. Information returned by the SNAPBP administrative view and the SNAP_GET_BP table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------|-----------|--|
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPBP_PART administrative view and SNAP_GET_BP_PART table function - Retrieve bufferpool_nodeinfo logical data group snapshot information

The SNAPBP_PART administrative view and the SNAP_GET_BP_PART table function return information about buffer pools from a bufferpool snapshot, in particular, the bufferpool_nodeinfo logical data group.

Important: The SNAPBP_PART administrative view and the SNAP_GET_BP_PART table function are deprecated and have been replaced by the “MON_GET_BUFFERPOOL table function - Get buffer pool metrics” on page 571.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPBP_PART administrative view”
- “SNAP_GET_BP_PART table function” on page 1294

SNAPBP_PART administrative view

This administrative view allows you to retrieve bufferpool_nodeinfo logical data group snapshot information for the currently connected database.

Used with the MON_GET_BUFFERPOOL table function, the SNAPBP_PART administrative view provides the data equivalent to the **GET SNAPSHOT FOR BUFFERPOOLS ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 281 on page 1296 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPBP_PART administrative view
- CONTROL privilege on the SNAPBP_PART administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_BP_PART table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMOINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve data for all bufferpools when connected to SAMPLE database.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, SUBSTR(BP_NAME,1,15) AS BP_NAME,
       BP_CUR_BUFSZ, BP_NEW_BUFSZ, BP_PAGES_LEFT_TO_REMOVE, BP_TBSP_USE_COUNT
FROM SYSIBMADM.SNAPBP_PART
```

The following is an example of output from this query.

| DB_NAME | BP_NAME | BP_CUR_BUFSZ | BP_NEW_BUFSZ | ... |
|---------|----------------|--------------|--------------|-----|
| SAMPLE | IBMDEFAULTBP | 1000 | 1000 | ... |
| SAMPLE | IBMSYSTEMBP4K | 16 | 16 | ... |
| SAMPLE | IBMSYSTEMBP8K | 16 | 16 | ... |
| SAMPLE | IBMSYSTEMBP16K | 16 | 16 | ... |

4 record(s) selected.

Output from this query (continued).

| ... | BP_PAGES_LEFT_TO_REMOVE | BP_TBSP_USE_COUNT |
|-----|-------------------------|-------------------|
| ... | 0 | 3 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | | |

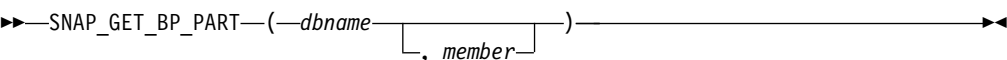
SNAP_GET_BP_PART table function

The SNAP_GET_BP_PART table function returns the same information as the SNAPBP_PART administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_BP table function, the SNAP_GET_BP_PART table function provides the data equivalent to the **GET SNAPSHOT FOR ALL BUFFERPOOLS** CLP command.

Refer to Table 281 on page 1296 for a complete list of information that can be returned.

Syntax



The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot for all bufferpools in all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_BP_PART table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_BP_PART table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve data for all bufferpools for all active databases when connected to the SAMPLE database.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, SUBSTR(BP_NAME,1,15) AS BP_NAME,  
       BP_CUR_BUFSZ, BP_NEW_BUFSZ, BP_PAGES_LEFT_TO_REMOVE, BP_TBSP_USE_COUNT  
FROM TABLE(SNAP_GET_BP_PART(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

| DB_NAME | BP_NAME | BP_CUR_BUFSZ | BP_NEW_BUFSZ | ... |
|---------|----------------|--------------|--------------|-----|
| SAMPLE | IBMDEFAULTBP | 250 | 250 | ... |
| SAMPLE | IBMSYSTEMBP4K | 16 | 16 | ... |
| SAMPLE | IBMSYSTEMBP8K | 16 | 16 | ... |
| SAMPLE | IBMSYSTEMBP16K | 16 | 16 | ... |
| SAMPLE | IBMSYSTEMBP32K | 16 | 16 | ... |
| TESTDB | IBMDEFAULTBP | 250 | 250 | ... |
| TESTDB | IBMSYSTEMBP4K | 16 | 16 | ... |
| TESTDB | IBMSYSTEMBP8K | 16 | 16 | ... |
| TESTDB | IBMSYSTEMBP16K | 16 | 16 | ... |
| TESTDB | IBMSYSTEMBP32K | 16 | 16 | ... |

...

Output from this query (continued).

... BP_PAGES_LEFT_TO_REMOVE BP_TBSP_USE_COUNT

| ... | BP_PAGES_LEFT_TO_REMOVE | BP_TBSP_USE_COUNT |
|-----|-------------------------|-------------------|
| ... | 0 | 3 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 3 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |

...

Information returned

Table 281. Information returned by the SNAPBP_PART administrative view and the SNAP_GET_BP_PART table function

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| BP_NAME | VARCHAR(128) | bp_name - Buffer pool name |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| BP_CUR_BUFSZ | BIGINT | bp_cur_bufsz - current size of buffer pool |
| BP_NEW_BUFSZ | BIGINT | bp_new_bufsz - New buffer pool size |
| BP_PAGES_LEFT_TO_REMOVE | BIGINT | bp_pages_left_to_remove - Number of pages left to remove |
| BP_TBSP_USE_COUNT | BIGINT | bp_tbsp_use_count - Number of table spaces mapped to buffer pool |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAP_GET_BP_V95 table function - Retrieve bufferpool logical group snapshot information

The SNAP_GET_BP_V95 table function returns information about buffer pools from a bufferpool snapshot, in particular, the bufferpool logical data group.

Important: The SNAP_GET_BP_V95 table function is deprecated and have been replaced by the "MON_GET_BUFFERPOOL table function - Get buffer pool metrics" on page 571.

Refer to Table 282 on page 1298 for a complete list of information that can be returned.

Syntax

```
»» SNAP_GET_BP_V95 ( ( dbname [ , member ] ) )
```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_BP_V95 table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_BP_V95 table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve total physical and logical reads for all bufferpools for all active databases for the currently connected database member.

```
SELECT SUBSTR(T.DB_NAME,1,10) AS DB_NAME,
SUBSTR(T.BP_NAME,1,20) AS BP_NAME,
(T.POOL_DATA_L_READS+T.POOL_INDEX_L_READS) AS TOTAL_LOGICAL_READS,
(T.POOL_DATA_P_READS+T.POOL_INDEX_P_READS) AS TOTAL_PHYSICAL_READS,
T.DBPARTITIONNUM
FROM TABLE(SNAP_GET_BP_V95(CAST(NULL AS VARCHAR(128)), -1)) AS T
```

The following is an example of output from this query.

```
DB_NAME    BP_NAME    TOTAL_LOGICAL_READS  ...
-----
SAMPLE     IBMDEFAULTBP          0 ...
TOOLSDB    IBMDEFAULTBP          0 ...
TOOLSDB    BP32K0000            0 ...
3 record(s) selected.
```

Output from this query (continued).

```
... TOTAL_PHYSICAL_READS DBPARTITIONNUM
... -----
...                0                0
...                0                0
...                0                0
```

Information returned

Table 282. Information returned by the SNAP_GET_BP_V95 table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| BP_NAME | VARCHAR(128) | bp_name - Buffer pool name |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| DB_PATH | VARCHAR(1024) | db_path - Database path |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input database alias |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |

Table 282. Information returned by the SNAP_GET_BP_V95 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|---|
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes |
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer pool asynchronous read time |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |

Table 282. Information returned by the SNAP_GET_BP_V95 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| FILES_CLOSED | BIGINT | files_closed - Database files closed |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer pool no victim buffers |
| PAGES_FROM_BLOCK_IOS | BIGINT | pages_from_block_ios - Total number of pages read by block I/O |
| PAGES_FROM_VECTORED_IOS | BIGINT | pages_from_vectored_ios - Total pages read by vectored I/O |
| VECTORED_IOS | BIGINT | vectored_ios - Number of vectored I/O requests |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

SNAPCONTAINER administrative view and SNAP_GET_CONTAINER table function - Retrieve tablespace_container logical data group snapshot information

The SNAPCONTAINER administrative view and the SNAP_GET_CONTAINER table function return table space snapshot information from the tablespace_container logical data group.

Important: The SNAPCONTAINER administrative view and the SNAP_GET_CONTAINER table function are deprecated and have been replaced by the “MON_GET_CONTAINER table function - Get table space container metrics” on page 616.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPCONTAINER administrative view”
- “SNAP_GET_CONTAINER table function” on page 1302

SNAPCONTAINER administrative view

This administrative view allows you to retrieve tablespace_container logical data group snapshot information for the currently connected database.

Used with the MON_GET_TABLESPACE table function and the SNAPTBSP_QUIESCER SNAPTBSP_RANGE administrative view, the SNAPCONTAINER administrative view returns data equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 283 on page 1304 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPCONTAINER administrative view
- CONTROL privilege on the SNAPCONTAINER administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_CONTAINER table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve details for the table space containers for all database members for the currently connected database.

```
SELECT SNAPSHOT_TIMESTAMP, SUBSTR(TBSP_NAME, 1, 15) AS TBSP_NAME,  
       TBSP_ID, SUBSTR(CONTAINER_NAME, 1, 20) AS CONTAINER_NAME,  
       CONTAINER_ID, CONTAINER_TYPE, ACCESSIBLE, DBPARTITIONNUM  
FROM SYSIBMADM.SNAPCONTAINER ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | TBSP_NAME | TBSP_ID | ... |
|----------------------------|--------------|---------|-----|
| 2006-01-08-16.49.24.639945 | SYSCATSPACE | 0 | ... |
| 2006-01-08-16.49.24.639945 | TEMPSPACE1 | 1 | ... |
| 2006-01-08-16.49.24.639945 | USERSPACE1 | 2 | ... |
| 2006-01-08-16.49.24.639945 | SYSTOOLSPACE | 3 | ... |
| 2006-01-08-16.49.24.640747 | TEMPSPACE1 | 1 | ... |
| 2006-01-08-16.49.24.640747 | USERSPACE1 | 2 | ... |
| 2006-01-08-16.49.24.639981 | TEMPSPACE1 | 1 | ... |
| 2006-01-08-16.49.24.639981 | USERSPACE1 | 2 | ... |
| | | | ... |

8 record(s) selected.

Output from this query (continued).

| ... | CONTAINER_NAME | CONTAINER_ID | CONTAINER_TYPE | ... |
|-----|----------------------|--------------|-----------------|-----|
| ... | ... | ... | ... | ... |
| ... | /home/swalkty/swalkt | 0 | FILE_EXTENT_TAG | ... |
| ... | /home/swalkty/swalkt | 0 | PATH | ... |
| ... | /home/swalkty/swalkt | 0 | FILE_EXTENT_TAG | ... |
| ... | /home/swalkty/swalkt | 0 | FILE_EXTENT_TAG | ... |
| ... | /home/swalkty/swalkt | 0 | PATH | ... |
| ... | /home/swalkty/swalkt | 0 | FILE_EXTENT_TAG | ... |
| ... | /home/swalkty/swalkt | 0 | PATH | ... |
| ... | /home/swalkty/swalkt | 0 | FILE_EXTENT_TAG | ... |

Output from this query (continued).

| ... | ACCESSIBLE | DBPARTITIONNUM |
|-----|------------|----------------|
| ... | 1 | 0 |
| ... | 1 | 0 |
| ... | 1 | 0 |
| ... | 1 | 0 |
| ... | 1 | 1 |
| ... | 1 | 1 |
| ... | 1 | 2 |
| ... | 1 | 2 |

SNAP_GET_CONTAINER table function

The SNAP_GET_CONTAINER table function returns the same information as the SNAPCONTAINER administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_TBSP, SNAP_GET_TBSP_PART, SNAP_GET_TBSP_QUIESCER and SNAP_GET_TBSP_RANGE table functions, the SNAP_GET_CONTAINER table function returns data equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 283 on page 1304 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_CONTAINER ((*dbname* [, *member*])) ►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all members where the database is active.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_CONTAINER table function takes a snapshot for the currently connected database and database member.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_CONTAINER table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve details for the table space containers on the currently connected database on the currently connected database member.

```

SELECT SNAPSHOT_TIMESTAMP, TBSP_NAME, TBSP_ID, CONTAINER_NAME,
       CONTAINER_ID, CONTAINER_TYPE, ACCESSIBLE
FROM TABLE(SNAP_GET_CONTAINER(' ', -1)) AS T

```

The following is an example of output from this query.

```

SNAPSHOT_TIMESTAMP      TBSP_NAME      TBSP_ID ...
-----
2005-04-25-14.42.10.899253 SYSCATSPACE      0 ...
2005-04-25-14.42.10.899253 TEMPSPACE1        1 ...
2005-04-25-14.42.10.899253 USERSPACE1        2 ...
2005-04-25-14.42.10.899253 SYSTOOLSPACE      3 ...
2005-04-25-14.42.10.899253 MYTEMP            4 ...
2005-04-25-14.42.10.899253 WHATSNEWTMPSPACE  5 ...

```

Output from this query (continued).

```

... CONTAINER_NAME      CONTAINER_ID ...
-----
... D:\DB2\NODE0000\SQL00002\SQLT0000.0      0 ...
... D:\DB2\NODE0000\SQL00002\SQLT0001.0      0 ...
... D:\DB2\NODE0000\SQL00002\SQLT0002.0      0 ...
... D:\DB2\NODE0000\SQL00002\SYSTOOLSPACE      0 ...
... D:\DB2\NODE0000\SQL003                    0 ...
... d:\DGTsWhatsNewContainer                  0 ...

```

Output from this query (continued).

```

... CONTAINER_TYPE ACCESSIBLE
... -----
... CONT_PATH      1
... CONT_PATH      1
... CONT_PATH      1
... CONT_PATH      1
... CONT_PATH      1
... CONT_PATH      1

```

Information returned

NOTE: The BUFFERPOOL database manager monitor switch must be turned on in order for the file system information to be returned.

Table 283. Information returned by the SNAPCONTAINER administrative view and the SNAP_GET_CONTAINER table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container name |
| CONTAINER_ID | BIGINT | container_id - Container identification |

Table 283. Information returned by the SNAPCONTAINER administrative view and the SNAP_GET_CONTAINER table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------|-------------|--|
| CONTAINER_TYPE | VARCHAR(16) | container_type - Container type. This is a text identifier based on the defines in sqlutil.h and is one of: <ul style="list-style-type: none"> • DISK_EXTENT_TAG • DISK_PAGE_TAG • FILE_EXTENT_TAG • FILE_PAGE_TAG • PATH |
| TOTAL_PAGES | BIGINT | container_total_pages - Total pages in container |
| USABLE_PAGES | BIGINT | container_usable_pages - Usable pages in container |
| ACCESSIBLE | SMALLINT | container_accessible - Accessibility of container |
| STRIPE_SET | BIGINT | container_stripe_set - Stripe set |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| FS_ID | VARCHAR(22) | fs_id - Unique file system identification number |
| FS_TOTAL_SIZE | BIGINT | fs_total_size - Total size of a file system |
| FS_USED_SIZE | BIGINT | fs_used_size - Amount of space used on a file system |

SNAP_GET_CONTAINER_V91 table function - Retrieve tablespace_container logical data group snapshot information

SNAP_GET_CONTAINER_V91 returns table space snapshot information from the tablespace_container logical data group.

Note: This table function has been deprecated and replaced by the SNAPCONTAINER administrative view and SNAP_GET_CONTAINER table function

Refer to Table 284 on page 1307 for a complete list of information that can be returned.

Syntax

```
►► SNAP_GET_CONTAINER_V91 ( (—dbname—, dbpartitionnum) ) ►►
```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

dbpartitionnum

An optional input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for an aggregate of all active database partitions. If *dbname* is not set to NULL and *dbpartitionnum* is set to NULL, -1 is set implicitly for *dbpartitionnum*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database partitions. An active database partition is a partition where the database is available for connection and use by applications.

If both *dbname* and *dbpartitionnum* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_CONTAINER_V91 table function takes a snapshot for the currently connected database and database partition number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_CONTAINER_V91 table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve details for the table space containers on the currently connected database on the currently connected database partition.

```
SELECT SNAPSHOT_TIMESTAMP, TBSP_NAME, TBSP_ID, CONTAINER_NAME,  
       CONTAINER_ID, CONTAINER_TYPE, ACCESSIBLE  
FROM TABLE(SNAP_GET_CONTAINER_V91(' ', -1)) AS T
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | TBSP_NAME | TBSP_ID | ... |
|----------------------------|-------------|---------|-----|
| 2005-04-25-14.42.10.899253 | SYSCATSPACE | 0 | ... |
| 2005-04-25-14.42.10.899253 | TEMPSPACE1 | 1 | ... |

```

2005-04-25-14.42.10.899253 USERSPACE1          2 ...
2005-04-25-14.42.10.899253 SYSTOOLSPACE         3 ...
2005-04-25-14.42.10.899253 MYTEMP              4 ...
2005-04-25-14.42.10.899253 WHATSNEWTEMPSPACE 5 ...

```

Output from this query (continued).

```

... CONTAINER_NAME                                CONTAINER_ID ...
... -----
... D:\DB2\NODE0000\SQL00002\SQLT0000.0          0 ...
... D:\DB2\NODE0000\SQL00002\SQLT0001.0          0 ...
... D:\DB2\NODE0000\SQL00002\SQLT0002.0          0 ...
... D:\DB2\NODE0000\SQL00002\SYSTOOLSPACE         0 ...
... D:\DB2\NODE0000\SQL003                        0 ...
... d:\DGTTsWhatsNewContainer                    0 ...

```

Output from this query (continued).

```

... CONTAINER_TYPE ACCESSIBLE
... -----
... CONT_PATH          1
... CONT_PATH          1
... CONT_PATH          1
... CONT_PATH          1
... CONT_PATH          1
... CONT_PATH          1

```

Information returned

NOTE: The **bufferpool** database manager monitor switch must be turned on in order for the file system information to be returned.

Table 284. Information returned by the `SNAP_GET_CONTAINER_V91` table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container name |
| CONTAINER_ID | BIGINT | container_id - Container identification |
| CONTAINER_TYPE | VARCHAR(16) | container_type - Container type. This is a text identifier based on the defines in <code>sqlutil.h</code> and is one of: <ul style="list-style-type: none"> • DISK_EXTENT_TAG • DISK_PAGE_TAG • FILE_EXTENT_TAG • FILE_PAGE_TAG • PATH |
| TOTAL_PAGES | BIGINT | container_total_pages - Total pages in container |
| USABLE_PAGES | BIGINT | container_usable_pages - Usable pages in container |

Table 284. Information returned by the SNAP_GET_CONTAINER_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------|-------------|--|
| ACCESSIBLE | SMALLINT | container_accessible - Accessibility of container |
| STRIPE_SET | BIGINT | container_stripe_set - Stripe set |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| FS_ID | VARCHAR(22) | fs_id - Unique file system identification number |
| FS_TOTAL_SIZE | BIGINT | fs_total_size - Total size of a file system |
| FS_USED_SIZE | BIGINT | fs_used_size - Amount of space used on a file system |

SNAPDB administrative view and SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group

The SNAPDB administrative view and the SNAP_GET_DB table function return snapshot information from the database (dbase) logical group.

Important: The SNAPDB administrative view and SNAP_GET_DB table function are deprecated and have been replaced by the “MON_GET_DATABASE table function - Get database metrics” on page 619 and “MON_GET_TRANSACTION_LOG table function - Get log information” on page 827.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPDB administrative view”
- “SNAP_GET_DB table function” on page 1310

SNAPDB administrative view

This administrative view allows you to retrieve snapshot information from the dbase logical group for the currently connected database.

Used in conjunction with ADMIN_GET_STORAGE_PATHS table function, MON_GET_MEMORY_POOL, MON_GET_TRANSACTION_LOG, and MON_GET_HADR, the SNAPDB administrative view provides information equivalent to the **GET SNAPSHOT FOR DATABASE on database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 285 on page 1312 for a complete list of information that is returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPDB administrative view

- CONTROL privilege on the SNAPDB administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DB table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Examples

Retrieve the status, platform, location, and connect time for all database members of the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME, DB_STATUS, SERVER_PLATFORM,
       DB_LOCATION, DB_CONN_TIME, DBPARTITIONNUM
FROM SYSIBMADM.SNAPDB ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| DB_NAME | DB_STATUS | SERVER_PLATFORM | DB_LOCATION | ... |
|---------|-----------|-----------------|-------------|-----|
| TEST | ACTIVE | AIX64 | LOCAL | ... |
| TEST | ACTIVE | AIX64 | LOCAL | ... |
| TEST | ACTIVE | AIX64 | LOCAL | ... |

3 record(s) selected.

Output from this query (continued).

| ... | DB_CONN_TIME | DBPARTITIONNUM |
|-----|----------------------------|----------------|
| ... | 2006-01-08-16.48.30.665477 | 0 |
| ... | 2006-01-08-16.48.34.005328 | 1 |
| ... | 2006-01-08-16.48.34.007937 | 2 |

This routine can be used by calling the following on the command line:

```
SELECT TOTAL_OLAP_FUNCS, OLAP_FUNC_OVERFLOWS, ACTIVE_OLAP_FUNCS
FROM SYSIBMADM.SNAPDB
```

| TOTAL_OLAP_FUNCS | OLAP_FUNC_OVERFLOWS | ACTIVE_OLAP_FUNCS |
|------------------|---------------------|-------------------|
| 7 | 2 | 1 |

1 record(s) selected.

After running a workload, a user can use the following query:

```
SELECT STATS_CACHE_SIZE, STATS_FABRICATIONS, SYNC_RUNSTATS,
       ASYNC_RUNSTATS, STATS_FABRICATE_TIME, SYNC_RUNSTATS_TIME
FROM SYSIBMADM.SNAPDB

STATS_CACHE_SIZE STATS_FABRICATIONS SYNC_RUNSTATS ASYNC_RUNSTATS ...
-----
128                2                1                0 ...

... STATS_FABRICATE_TIME SYNC_RUNSTATS_TIME
... -----
...                10                100
```

1 record(s) selected.

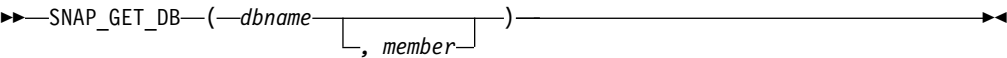
SNAP_GET_DB table function

The SNAP_GET_DB table function returns the same information as the SNAPDB administrative view.

Used in conjunction with the ADMIN_GET_STORAGE_PATHS table function, MON_GET_MEMORY_POOL, MON_GET_TRANSACTION_LOG, and MON_GET_HADR table functions, the SNAP_GET_DB table function provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES** CLP command.

Refer to Table 285 on page 1312 for a complete list of information that is returned.

Syntax



The schema is SYSPROC.

Table function parameters

dbname
An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member
An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file

with the corresponding snapshot API request type does not exist, then the SNAP_GET_DB table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DB table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Retrieve the status, platform, location, and connect time as an aggregate view across all database members of the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME, DB_STATUS, SERVER_PLATFORM,
       DB_LOCATION, DB_CONN_TIME FROM TABLE(SNAP_GET_DB(' ', -2)) AS T
```

The following is an example of output from this query.

| DB_NAME | DB_STATUS | SERVER_PLATFORM | ... |
|-----------|-----------|-----------------|-----|
| -----...- | -----...- | -----...- | ... |
| SAMPLE | ACTIVE | AIX64 | ... |

1 record(s) selected.

Output from this query (continued).

| ... DB_LOCATION | DB_CONN_TIME |
|-----------------|----------------------------|
| ... ----- | ----- |
| ... LOCAL | 2005-07-24-22.09.22.013196 |

Example 2: Retrieve the status, platform, location, and connect time as an aggregate view across all database members for all active databases in the same instance that contains the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME, DB_STATUS, SERVER_PLATFORM,
       DB_LOCATION, DB_CONN_TIME
FROM TABLE(SNAP_GET_DB(CAST (NULL AS VARCHAR(128)), -2)) AS T
```

The following is an example of output from this query.

| DB_NAME | DB_STATUS | SERVER_PLATFORM | ... |
|-----------|-----------|-----------------|-----|
| -----...- | -----...- | -----...- | ... |
| TOOLSDB | ACTIVE | AIX64 | ... |
| SAMPLE | ACTIVE | AIX64 | ... |

Output from this query (continued).

```

... DB_LOCATION DB_CONN_TIME
... -----
... LOCAL      2005-07-24-22.26.54.396335
... LOCAL      2005-07-24-22.09.22.013196

```

Example 3: This routine can be used by calling the following on the command line:

When connected to a database:

```

SELECT TOTAL_OLAP_FUNCS, OLAP_FUNC_OVERFLOWS, ACTIVE_OLAP_FUNCS
FROM TABLE (SNAP_GET_DB('', 0)) AS T

```

The output will look like:

```

TOTAL_OLAP_FUNCS OLAP_FUNC_OVERFLOWS ACTIVE_OLAP_FUNCS
-----
              7                2                1

```

1 record(s) selected.

Example 4: After running a workload, a user can use the following query with the table function.

```

SELECT STATS_CACHE_SIZE, STATS_FABRICATIONS, SYNC_RUNSTATS,
       ASYNC_RUNSTATS, STATS_FABRICATE_TIME, SYNC_RUNSTATS_TIME
FROM TABLE (SNAP_GET_DB('mytestdb', -1)) AS SNAPDB

```

```

STATS_CACHE_SIZE STATS_FABRICATIONS SYNC_RUNSTATS ASYNC_RUNSTATS ...
-----
              200                1                2                0 ...

```

Continued

```

...STATS_FABRICATE_TIME SYNC_RUNSTATS_TIME
-----
...                2                32

```

1 record(s) selected.

Example 5: The following example shows how you can use the SNAP_GET_DB table function to determine the status of a database:

```

SELECT SUBSTR
       (DB_NAME, 1, 20) AS DB_NAME, DB_STATUS
FROM table(SNAP_GET_DB('hadrdb', 0))

```

```

DB_NAME          DB_STATUS
-----
HADRDB           ACTIVE_STANDBY

```

Information returned

Table 285. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|---------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| DB_PATH | VARCHAR(1024) | db_path - Database path |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input database alias |

Table 285. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|--------------|---|
| DB_STATUS | VARCHAR(16) | db_status - Status of database. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • ACTIVE • QUIESCE_PEND • QUIESCED • ROLLFWD • ACTIVE_STANDBY - the HADR database is in a standby mode with reads on standby enabled. • STANDBY - the HADR database is in standby mode (reads on standby are not enabled). |
| CATALOG_PARTITION | SMALLINT | catalog_node - Catalog database partition number |
| CATALOG_PARTITION_NAME | VARCHAR(128) | catalog_node_name - Catalog database partition network name |

Table 285. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------|-------------|---|
| SERVER_PLATFORM | VARCHAR(12) | <p>server_platform - Server operating system. This interface returns a text identifier based on defines in sqlmon.h, and is one of:</p> <ul style="list-style-type: none"> • AIX • AIX64 • AS400_DRDA • DOS • DYNIX • HP • HP64 • HPIA • HPIA64 • LINUX • LINUX390 • LINUXIA64 • LINUXPPC • LINUXPPC64 • LINUXX8664 • LINUXZ64 • MAC • MVS_DRDA • NT • NT64 • OS2 • OS390 • SCO • SGI • SNI • SUN • SUN64 • UNKNOWN • UNKNOWN_DRDA • VM_DRDA • VSE_DRDA • WINDOWS |
| DB_LOCATION | VARCHAR(12) | <p>db_location - Database location. This interface returns a text identifier based on defines in sqlmon.h, and is one of:</p> <ul style="list-style-type: none"> • LOCAL • REMOTE |
| DB_CONN_TIME | TIMESTAMP | db_conn_time - Database activation timestamp |
| LAST_RESET | TIMESTAMP | last_reset - Last reset timestamp |
| LAST_BACKUP | TIMESTAMP | last_backup - Last backup timestamp |

Table 285. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| CONNECTIONS_TOP | BIGINT | connections_top - Maximum number of concurrent connections |
| TOTAL_CONS | BIGINT | total_cons - Connects since database activation |
| TOTAL_SEC_CONS | BIGINT | total_sec_cons - Secondary connections |
| APPLS_CUR_CONS | BIGINT | appls_cur_cons - Applications connected currently |
| APPLS_IN_DB2 | BIGINT | appls_in_db2 - Applications executing in the database currently |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of associated agents |
| AGENTS_TOP | BIGINT | agents_top - Number of agents created |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum number of coordinating agents |
| LOCKS_HELD | BIGINT | locks_held - Locks held |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_LIST_IN_USE | BIGINT | lock_list_in_use - Total lock list memory in use |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive lock escalations |
| LOCKS_WAITING | BIGINT | locks_waiting - Current agents waiting on locks |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |
| NUM_INDOUBT_TRANS | BIGINT | num_indoubt_trans - Number of indoubt transactions |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total sort heap allocated |
| SORT_SHRHEAP_ALLOCATED | BIGINT | sort_shrheap_allocated - Sort share heap currently allocated |
| SORT_SHRHEAP_TOP | BIGINT | sort_shrheap_top - Sort share heap high water mark |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post shared threshold sorts |
| TOTAL_SORTS | BIGINT | total_sorts - Total sorts |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total sort time |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| ACTIVE_SORTS | BIGINT | active_sorts - Active sorts |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |

Table 285. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer pool asynchronous read time |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time |

Table 285. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|---|
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests |
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer pool no victim buffers |
| POOL_LSN_GAP_CLNS | BIGINT | pool_lsn_gap_clns - Buffer pool log space cleaners triggered |
| POOL_DRTY_PG_STEAL_CLNS | BIGINT | pool_drtz_pg_steal_clns - Buffer pool victim page cleaners triggered |
| POOL_DRTY_PG_THRSH_CLNS | BIGINT | pool_drtz_pg_thrsh_clns - Buffer pool threshold cleaners triggered |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| FILES_CLOSED | BIGINT | files_closed - Database files closed |
| ELAPSED_EXEC_TIME_S | BIGINT | elapsed_exec_time - Statement execution elapsed time |
| ELAPSED_EXEC_TIME_MS | BIGINT | elapsed_exec_time - Statement execution elapsed time |
| COMMIT_SQL_STMTS | BIGINT | commit_sql_stmts - Commit statements attempted |
| ROLLBACK_SQL_STMTS | BIGINT | rollback_sql_stmts - Rollback statements attempted |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL statements attempted |
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL statements attempted |
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed statement operations |
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL statements executed |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - UPDATE/INSERT/DELETE SQL statements executed |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data definition language (DDL) SQL statements |
| INT_AUTO_REBINDS | BIGINT | int_auto_rebinds - Internal automatic rebinds |

Table 285. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|-----------|--|
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal rows deleted |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal rows inserted |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal rows updated |
| INT_COMMITS | BIGINT | int_commits - Internal commits |
| INT_ROLLBACKS | BIGINT | int_rollbacks - Internal rollbacks |
| INT_DEADLOCK_ROLLBACKS | BIGINT | int_deadlock_rollbacks - Internal rollbacks due to deadlock |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated |
| ROWS_SELECTED | BIGINT | rows_selected - Rows selected |
| ROWS_READ | BIGINT | rows_read - Rows read |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/precompiles attempted |
| TOTAL_LOG_AVAILABLE | BIGINT | total_log_available - Total log available |
| TOTAL_LOG_USED | BIGINT | total_log_used - Total log space used |
| SEC_LOG_USED_TOP | BIGINT | sec_log_used_top - Maximum secondary log space used |
| TOT_LOG_USED_TOP | BIGINT | tot_log_used_top - Maximum total log space used |
| SEC_LOGS_ALLOCATED | BIGINT | sec_logs_allocated - Secondary logs allocated currently |
| LOG_READS | BIGINT | log_reads - Number of log pages read |
| LOG_READ_TIME_S | BIGINT | log_read_time - Log read time |
| LOG_READ_TIME_NS | BIGINT | log_read_time - Log read time |
| LOG_WRITES | BIGINT | log_writes - Number of log pages written |
| LOG_WRITE_TIME_S | BIGINT | log_write_time - Log write time |
| LOG_WRITE_TIME_NS | BIGINT | log_write_time - Log write time |
| NUM_LOG_WRITE_IO | BIGINT | num_log_write_io - Number of log writes |
| NUM_LOG_READ_IO | BIGINT | num_log_read_io - Number of log reads |
| NUM_LOG_PART_PAGE_IO | BIGINT | num_log_part_page_io - Number of partial log page writes |
| NUM_LOG_BUFFER_FULL | BIGINT | num_log_buffer_full - Number of full log buffers |
| NUM_LOG_DATA_FOUND_IN_BUFFER | BIGINT | num_log_data_found_in_buffer - Number of log data found in buffer |
| APPL_ID_OLDEST_XACT | BIGINT | appl_id_oldest_xact - Application with oldest transaction |
| LOG_TO_REDO_FOR_RECOVERY | BIGINT | log_to_redo_for_recovery - Amount of log to be redone for recovery |
| LOG_HELD_BY_DIRTY_PAGES | BIGINT | log_held_by_dirty_pages - Amount of log space accounted for by dirty pages |

Table 285. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------------|-----------|--|
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts |
| PKG_CACHE_NUM_OVERFLOWS | BIGINT | pkg_cache_num_overflows - Package cache overflows |
| PKG_CACHE_SIZE_TOP | BIGINT | pkg_cache_size_top - Package cache high water mark |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - Section lookups |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Section inserts |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog cache overflows |
| CAT_CACHE_SIZE_TOP | BIGINT | cat_cache_size_top - Catalog cache high water mark |
| PRIV_WORKSPACE_SIZE_TOP | BIGINT | priv_workspace_size_top - Maximum private workspace size |
| PRIV_WORKSPACE_NUM_OVERFLOWS | BIGINT | priv_workspace_num_overflows - Private workspace overflows |
| PRIV_WORKSPACE_SECTION_INSERTS | BIGINT | priv_workspace_section_inserts - Private workspace section inserts |
| PRIV_WORKSPACE_SECTION_LOOKUPS | BIGINT | priv_workspace_section_lookups - Private workspace section lookups |
| SHR_WORKSPACE_SIZE_TOP | BIGINT | shr_workspace_size_top - Maximum shared workspace size |
| SHR_WORKSPACE_NUM_OVERFLOWS | BIGINT | shr_workspace_num_overflows - Shared workspace overflows |
| SHR_WORKSPACE_SECTION_INSERTS | BIGINT | shr_workspace_section_inserts - Shared workspace section inserts |
| SHR_WORKSPACE_SECTION_LOOKUPS | BIGINT | shr_workspace_section_lookups - Shared workspace section lookups |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total hash joins |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total hash loops |
| HASH_JOIN_OVERFLOWS | BIGINT | hash_join_overflows - Hash join overflows |
| HASH_JOIN_SMALL_OVERFLOWS | BIGINT | hash_join_small_overflows - Hash join small overflows |
| POST_SHRTHRESHOLD_HASH_JOINS | BIGINT | post_shrthreshold_hash_joins - Post threshold hash joins |
| ACTIVE_HASH_JOINS | BIGINT | active_hash_joins - Active hash joins |
| NUM_DB_STORAGE_PATHS | BIGINT | num_db_storage_paths - Number of automatic storage paths |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| SMALLEST_LOG_AVAIL_NODE | INTEGER | smallest_log_avail_node - Node with least available log space |

Table 285. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|-----------|--|
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP functions |
| OLAP_FUNC_OVERFLOWS | BIGINT | olap_func_overflows - OLAP function overflows |
| ACTIVE_OLAP_FUNCS | BIGINT | active_olap_funcs - Active OLAP functions |
| STATS_CACHE_SIZE | BIGINT | stats_cache_size - Size of statistics cache |
| STATS_FABRICATIONS | BIGINT | stats_fabrications - Total number of statistics fabrications |
| SYNC_RUNSTATS | BIGINT | sync_runstats - Total number of synchronous RUNSTATS activities |
| ASYNC_RUNSTATS | BIGINT | async_runstats - Total number of asynchronous RUNSTATS requests |
| STATS_FABRICATE_TIME | BIGINT | stats_fabricate_time - Total time spent on statistics fabrication activities |
| SYNC_RUNSTATS_TIME | BIGINT | sync_runstats_time - Total time spent on synchronous RUNSTATS activities |
| NUM_THRESHOLD_VIOLATIONS | BIGINT | num_threshold_violations - Number of threshold violations |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPDB_MEMORY_POOL administrative view and SNAP_GET_DB_MEMORY_POOL table function - Retrieve database level memory usage information

The SNAPDB_MEMORY_POOL administrative view and the SNAP_GET_DB_MEMORY_POOL table function return information about memory usage at the database level for UNIX platforms only.

The SNAPDB_MEMORY_POOL administrative view and the SNAP_GET_DB_MEMORY_POOL table function return information about memory usage at the database level for UNIX platforms only.

Important: Starting in Version 9.7 Fix Pack 5, the SNAPDB_MEMORY_POOL administrative view and SNAP_GET_DB_MEMORY_POOL table function have been deprecated and replaced by the “MON_GET_MEMORY_POOL - get memory pool information” on page 679 and “MON_GET_MEMORY_SET - get memory set information” on page 682.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPDB_MEMORY_POOL administrative view”
- “SNAP_GET_DB_MEMORY_POOL table function” on page 1322

SNAPDB_MEMORY_POOL administrative view

This administrative view allows you to retrieve database level memory usage information for the currently connected database.

Used with the MON_GET_DATABASE, MON_GET_TRANSACTION_LOG, MON_GET_HADR, and ADMIN_GET_STORAGE_PATHS table functions, the SNAPDB_MEMORY_POOL administrative view provides information equivalent to the **GET SNAPSHOT FOR DATABASE ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 286 on page 1323 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- SELECT privilege on the SNAPDB_MEMORY_POOL administrative view
- CONTROL privilege on the SNAPDB_MEMORY_POOL administrative view
- DATAACCESS authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DB_MEMORY_POOL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of memory pools and their current size for the currently connected database, SAMPLE.

```
SELECT POOL_ID, POOL_CUR_SIZE FROM SYSIBMADM.SNAPDB_MEMORY_POOL
```

The following is an example of output from this query.

| POOL_ID | POOL_CUR_SIZE |
|---------------|---------------|
| UTILITY | 32768 |
| PACKAGE_CACHE | 475136 |
| CAT_CACHE | 65536 |
| BP | 2097152 |
| BP | 1081344 |
| BP | 540672 |
| BP | 278528 |
| BP | 147456 |
| BP | 81920 |
| LOCK_MGR | 294912 |
| DATABASE | 3833856 |
| OTHER | 0 |

12 record(s) selected.

SNAP_GET_DB_MEMORY_POOL table function

The SNAP_GET_DB_MEMORY_POOL table function returns the same information as the SNAPDB_MEMORY_POOL administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DB, SNAP_GET_DETAILLOG, SNAP_GET_HADR and ADMIN_GET_STORAGE_PATHS table functions, the SNAP_GET_DB_MEMORY_POOL table function provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES** CLP command.

Refer to Table 286 on page 1323 for a complete list of information that can be returned.

Syntax

```
►► SNAP_GET_DB_MEMORY_POOL (—dbname—  
                             [ , member ] )
```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DB_MEMORY_POOL table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DB_MEMORY_POOL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of memory pools and their current size for all databases.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, POOL_ID, POOL_CUR_SIZE
FROM TABLE(SNAPSHOT_GET_DB_MEMORY_POOL
(CAST(NULL AS VARCHAR(128)), -1)) AS T
```

The following is an example of output from this query.

| DB_NAME | POOL_ID | POOL_CUR_SIZE |
|---------|---------------|---------------|
| TESTDB | UTILITY | 65536 |
| TESTDB | PACKAGE_CACHE | 851968 |
| TESTDB | CAT_CACHE | 65536 |
| TESTDB | BP | 35913728 |
| TESTDB | BP | 589824 |
| TESTDB | BP | 327680 |
| TESTDB | BP | 196608 |
| TESTDB | BP | 131072 |
| TESTDB | SHARED_SORT | 65536 |
| TESTDB | LOCK_MGR | 10092544 |
| TESTDB | DATABASE | 4980736 |
| TESTDB | OTHER | 196608 |
| SAMPLE | UTILITY | 65536 |
| SAMPLE | PACKAGE_CACHE | 655360 |
| SAMPLE | CAT_CACHE | 131072 |
| SAMPLE | BP | 4325376 |
| SAMPLE | BP | 589824 |
| SAMPLE | BP | 327680 |
| SAMPLE | BP | 196608 |
| SAMPLE | BP | 131072 |
| SAMPLE | SHARED_SORT | 0 |
| SAMPLE | LOCK_MGR | 655360 |
| SAMPLE | DATABASE | 4653056 |
| SAMPLE | OTHER | 196608 |

24 record(s) selected.

Information returned

Table 286. Information returned by the SNAPDB_MEMORY_POOL administrative view and the SNAP_GET_DB_MEMORY_POOL table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |

Table 286. Information returned by the SNAPDB_MEMORY_POOL administrative view and the SNAP_GET_DB_MEMORY_POOL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------|-------------|---|
| POOL_ID | VARCHAR(14) | pool_id - Memory pool identifier. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • APP_GROUP • APPL_CONTROL • APPLICATION • BP • CAT_CACHE • DATABASE • DFM • FCMBP • IMPORT_POOL • LOCK_MGR • MONITOR • OTHER • PACKAGE_CACHE • QUERY • SHARED_SORT • SORT • STATEMENT • STATISTICS • UTILITY |
| POOL_SECONDARY_ID | VARCHAR(32) | pool_secondary_id - Memory pool secondary identifier |
| POOL_CUR_SIZE | BIGINT | pool_cur_size - Current size of memory pool |
| POOL_WATERMARK | BIGINT | pool_watermark - Memory pool watermark |
| POOL_CONFIG_SIZE | BIGINT | pool_config_size - Configured size of memory pool |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAP_GET_DB_V97 table function - Retrieve snapshot information from the dbase logical group

The SNAP_GET_DB_V97 table function returns snapshot information from the database (dbase) logical group.

Important: The SNAP_GET_DB_V97 table function is deprecated and has been replaced by the “MON_GET_DATABASE table function - Get database metrics” on page 619 and “MON_GET_TRANSACTION_LOG table function - Get log information” on page 827.

Refer to Table 287 on page 1327 for a complete list of information that is returned.

Syntax

►► SNAP_GET_DB_V97 ((*dbname* [, *member*])) ►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DB_V97 table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DB_V97 table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Retrieve the status, platform, location, and connect time as an aggregate view across all database members of the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME, DB_STATUS, SERVER_PLATFORM,
       DB_LOCATION, DB_CONN_TIME FROM TABLE(SNAP_GET_DB_V97('', -2)) AS T
```

The following is an example of output from this query.

| DB_NAME | DB_STATUS | SERVER_PLATFORM | ... |
|---------|-----------|-----------------|-----|
| ----- | ----- | ----- | ... |
| SAMPLE | ACTIVE | AIX64 | ... |

1 record(s) selected.

Output from this query (continued).

| ... | DB_LOCATION | DB_CONN_TIME |
|-----|-------------|----------------------------|
| ... | ----- | ----- |
| ... | LOCAL | 2005-07-24-22.09.22.013196 |

Example 2: Retrieve the status, platform, location, and connect time as an aggregate view across all database members for all active databases in the same instance that contains the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME, DB_STATUS, SERVER_PLATFORM,
       DB_LOCATION, DB_CONN_TIME
FROM TABLE(SNAP_GET_DB_V97(CAST (NULL AS VARCHAR(128)), -2)) AS T
```

The following is an example of output from this query.

| DB_NAME | DB_STATUS | SERVER_PLATFORM | ... |
|---------|-----------|-----------------|-----|
| ----- | ----- | ----- | ... |
| TOOLSDB | ACTIVE | AIX64 | ... |
| SAMPLE | ACTIVE | AIX64 | ... |

Output from this query (continued).

| ... | DB_LOCATION | DB_CONN_TIME |
|-----|-------------|----------------------------|
| ... | ----- | ----- |
| ... | LOCAL | 2005-07-24-22.26.54.396335 |
| ... | LOCAL | 2005-07-24-22.09.22.013196 |

Example 3: This routine can be used by calling the following on the command line:

When connected to a database:

```
SELECT TOTAL_OLAP_FUNCS, OLAP_FUNC_OVERFLOWS, ACTIVE_OLAP_FUNCS
FROM TABLE (SNAP_GET_DB_V97('', 0)) AS T
```

The output will look like:

| TOTAL_OLAP_FUNCS | OLAP_FUNC_OVERFLOWS | ACTIVE_OLAP_FUNCS |
|------------------|---------------------|-------------------|
| ----- | ----- | ----- |
| 7 | 2 | 1 |

1 record(s) selected.

Example 4: After running a workload, a user can use the following query with the table function.

```
SELECT STATS_CACHE_SIZE, STATS_FABRICATIONS, SYNC_RUNSTATS,
       ASYNC_RUNSTATS, STATS_FABRICATE_TIME, SYNC_RUNSTATS_TIME
FROM TABLE (SNAP_GET_DB_V97('mytestdb', -1)) AS SNAPDB
```

STATS_CACHE_SIZE STATS_FABRICATIONS SYNC_RUNSTATS ASYNC_RUNSTATS ...

```
-----
                200                1                2                0 ...
```

Continued

```
...STATS_FABRICATE_TIME  SYNC_RUNSTATS_TIME
-----
...                2                32
```

1 record(s) selected.

Example 5: The following example shows how you can use the SNAP_GET_DB_V97 table function to determine the status of a database:

```
SELECT SUBSTR
      (DB_NAME, 1, 20) AS DB_NAME, DB_STATUS
FROM   table(SNAP_GET_DB_V97('hadrdb', 0))
```

```
DB_NAME          DB_STATUS
-----
HADRDB          ACTIVE_STANDBY
```

Information returned

Table 287. Information returned by the SNAP_GET_DB_V97 table function

| Column name | Data type | Description or corresponding monitor element |
|------------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| DB_PATH | VARCHAR(1024) | db_path - Database path |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input database alias |
| DB_STATUS | VARCHAR(16) | db_status - Status of database. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • ACTIVE • QUIESCE_PEND • QUIESCED • ROLLFWD • ACTIVE_STANDBY - the HADR database is in a standby mode with reads on standby enabled. • STANDBY - the HADR database is in standby mode (reads on standby are not enabled). |
| CATALOG_PARTITION | SMALLINT | catalog_node - Catalog node number |
| CATALOG_PARTITION_NAME | VARCHAR(128) | catalog_node_name - Catalog node network name |

Table 287. Information returned by the SNAP_GET_DB_V97 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------|-------------|--|
| SERVER_PLATFORM | VARCHAR(12) | server_platform - Server operating system. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • AIX • AIX64 • AS400_DRDA • DOS • DYNIX • HP • HP64 • HPIA • HPIA64 • LINUX • LINUX390 • LINUXIA64 • LINUXPPC • LINUXPPC64 • LINUXX8664 • LINUXZ64 • MAC • MVS_DRDA • NT • NT64 • OS2 • OS390 • SCO • SGI • SNI • SUN • SUN64 • UNKNOWN • UNKNOWN_DRDA • VM_DRDA • VSE_DRDA • WINDOWS |
| DB_LOCATION | VARCHAR(12) | db_location - Database location. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • LOCAL • REMOTE |
| DB_CONN_TIME | TIMESTAMP | db_conn_time - Database activation timestamp |
| LAST_RESET | TIMESTAMP | last_reset - Last reset timestamp |
| LAST_BACKUP | TIMESTAMP | last_backup - Last backup timestamp |

Table 287. Information returned by the SNAP_GET_DB_V97 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| CONNECTIONS_TOP | BIGINT | connections_top - Maximum number of concurrent connections |
| TOTAL_CONS | BIGINT | total_cons - Connects since database activation |
| TOTAL_SEC_CONS | BIGINT | total_sec_cons - Secondary connections |
| APPLS_CUR_CONS | BIGINT | appls_cur_cons - Applications connected currently |
| APPLS_IN_DB2 | BIGINT | appls_in_db2 - Applications executing in the database currently |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of associated agents |
| AGENTS_TOP | BIGINT | agents_top - Number of agents created |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum number of coordinating agents |
| LOCKS_HELD | BIGINT | locks_held - Locks held |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_LIST_IN_USE | BIGINT | lock_list_in_use - Total lock list memory in use |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive lock escalations |
| LOCKS_WAITING | BIGINT | locks_waiting - Current agents waiting on locks |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |
| NUM_INDOUBT_TRANS | BIGINT | num_indoubt_trans - Number of indoubt transactions |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total sort heap allocated |
| SORT_SHRHEAP_ALLOCATED | BIGINT | sort_shrheap_allocated - Sort share heap currently allocated |
| SORT_SHRHEAP_TOP | BIGINT | sort_shrheap_top - Sort share heap high water mark |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post shared threshold sorts |
| TOTAL_SORTS | BIGINT | total_sorts - Total sorts |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total sort time |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| ACTIVE_SORTS | BIGINT | active_sorts - Active sorts |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |

Table 287. Information returned by the SNAP_GET_DB_V97 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer pool asynchronous read time |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time |

Table 287. Information returned by the SNAP_GET_DB_V97 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|---|
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests |
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer pool no victim buffers |
| POOL_LSN_GAP_CLNS | BIGINT | pool_lsn_gap_clns - Buffer pool log space cleaners triggered |
| POOL_DRTY_PG_STEAL_CLNS | BIGINT | pool_drty_pg_steal_clns - Buffer pool victim page cleaners triggered |
| POOL_DRTY_PG_THRSH_CLNS | BIGINT | pool_drty_pg_thrsh_clns - Buffer pool threshold cleaners triggered |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| FILES_CLOSED | BIGINT | files_closed - Database files closed |
| ELAPSED_EXEC_TIME_S | BIGINT | elapsed_exec_time - Statement execution elapsed time |
| ELAPSED_EXEC_TIME_MS | BIGINT | elapsed_exec_time - Statement execution elapsed time |
| COMMIT_SQL_STMTS | BIGINT | commit_sql_stmts - Commit statements attempted |
| ROLLBACK_SQL_STMTS | BIGINT | rollback_sql_stmts - Rollback statements attempted |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL statements attempted |
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL statements attempted |
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed statement operations |
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL statements executed |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - UPDATE/INSERT/DELETE SQL statements executed |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data definition language (DDL) SQL statements |
| INT_AUTO_REBINDS | BIGINT | int_auto_rebinds - Internal automatic rebinds |

Table 287. Information returned by the SNAP_GET_DB_V97 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|-----------|--|
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal rows deleted |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal rows inserted |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal rows updated |
| INT_COMMITS | BIGINT | int_commits - Internal commits |
| INT_ROLLBACKS | BIGINT | int_rollbacks - Internal rollbacks |
| INT_DEADLOCK_ROLLBACKS | BIGINT | int_deadlock_rollbacks - Internal rollbacks due to deadlock |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated |
| ROWS_SELECTED | BIGINT | rows_selected - Rows selected |
| ROWS_READ | BIGINT | rows_read - Rows read |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/precompiles attempted |
| TOTAL_LOG_AVAILABLE | BIGINT | total_log_available - Total log available |
| TOTAL_LOG_USED | BIGINT | total_log_used - Total log space used |
| SEC_LOG_USED_TOP | BIGINT | sec_log_used_top - Maximum secondary log space used |
| TOT_LOG_USED_TOP | BIGINT | tot_log_used_top - Maximum total log space used |
| SEC_LOGS_ALLOCATED | BIGINT | sec_logs_allocated - Secondary logs allocated currently |
| LOG_READS | BIGINT | log_reads - Number of log pages read |
| LOG_READ_TIME_S | BIGINT | log_read_time - Log read time |
| LOG_READ_TIME_NS | BIGINT | log_read_time - Log read time |
| LOG_WRITES | BIGINT | log_writes - Number of log pages written |
| LOG_WRITE_TIME_S | BIGINT | log_write_time - Log write time |
| LOG_WRITE_TIME_NS | BIGINT | log_write_time - Log write time |
| NUM_LOG_WRITE_IO | BIGINT | num_log_write_io - Number of log writes |
| NUM_LOG_READ_IO | BIGINT | num_log_read_io - Number of log reads |
| NUM_LOG_PART_PAGE_IO | BIGINT | num_log_part_page_io - Number of partial log page writes |
| NUM_LOG_BUFFER_FULL | BIGINT | num_log_buffer_full - Number of full log buffers |
| NUM_LOG_DATA_FOUND_IN_BUFFER | BIGINT | num_log_data_found_in_buffer - Number of log data found in buffer |
| APPL_ID_OLDEST_XACT | BIGINT | appl_id_oldest_xact - Application with oldest transaction |
| LOG_TO_REDO_FOR_RECOVERY | BIGINT | log_to_redo_for_recovery - Amount of log to be redone for recovery |
| LOG_HELD_BY_DIRTY_PAGES | BIGINT | log_held_by_dirty_pages - Amount of log space accounted for by dirty pages |

Table 287. Information returned by the SNAP_GET_DB_V97 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------------|-----------|--|
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts |
| PKG_CACHE_NUM_OVERFLOWS | BIGINT | pkg_cache_num_overflows - Package cache overflows |
| PKG_CACHE_SIZE_TOP | BIGINT | pkg_cache_size_top - Package cache high water mark |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - Section lookups |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Section inserts |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog cache overflows |
| CAT_CACHE_SIZE_TOP | BIGINT | cat_cache_size_top - Catalog cache high water mark |
| PRIV_WORKSPACE_SIZE_TOP | BIGINT | priv_workspace_size_top - Maximum private workspace size |
| PRIV_WORKSPACE_NUM_OVERFLOWS | BIGINT | priv_workspace_num_overflows - Private workspace overflows |
| PRIV_WORKSPACE_SECTION_INSERTS | BIGINT | priv_workspace_section_inserts - Private workspace section inserts |
| PRIV_WORKSPACE_SECTION_LOOKUPS | BIGINT | priv_workspace_section_lookups - Private workspace section lookups |
| SHR_WORKSPACE_SIZE_TOP | BIGINT | shr_workspace_size_top - Maximum shared workspace size |
| SHR_WORKSPACE_NUM_OVERFLOWS | BIGINT | shr_workspace_num_overflows - Shared workspace overflows |
| SHR_WORKSPACE_SECTION_INSERTS | BIGINT | shr_workspace_section_inserts - Shared workspace section inserts |
| SHR_WORKSPACE_SECTION_LOOKUPS | BIGINT | shr_workspace_section_lookups - Shared workspace section lookups |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total hash joins |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total hash loops |
| HASH_JOIN_OVERFLOWS | BIGINT | hash_join_overflows - Hash join overflows |
| HASH_JOIN_SMALL_OVERFLOWS | BIGINT | hash_join_small_overflows - Hash join small overflows |
| POST_SHRTHRESHOLD_HASH_JOINS | BIGINT | post_shrthreshold_hash_joins - Post threshold hash joins |
| ACTIVE_HASH_JOINS | BIGINT | active_hash_joins - Active hash joins |
| NUM_DB_STORAGE_PATHS | BIGINT | num_db_storage_paths - Number of automatic storage paths |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| SMALLEST_LOG_AVAIL_NODE | INTEGER | smallest_log_avail_node - Node with least available log space |

Table 287. Information returned by the SNAP_GET_DB_V97 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|-----------|--|
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP functions |
| OLAP_FUNC_OVERFLOWS | BIGINT | olap_func_overflows - OLAP function overflows |
| ACTIVE_OLAP_FUNCS | BIGINT | active_olap_funcs - Active OLAP functions |
| STATS_CACHE_SIZE | BIGINT | stats_cache_size Size of statistics cache |
| STATS_FABRICATIONS | BIGINT | stats_fabrications Total number of statistics fabrications |
| SYNC_RUNSTATS | BIGINT | sync_runstats Total number of synchronous RUNSTATS activities |
| ASYNC_RUNSTATS | BIGINT | async_runstats Total number of asynchronous RUNSTATS requests |
| STATS_FABRICATE_TIME | BIGINT | stats_fabricate_time Total time spent on statistics fabrication activities |
| SYNC_RUNSTATS_TIME | BIGINT | sync_runstats_time Total time spent on synchronous RUNSTATS activities |
| NUM_THRESHOLD_VIOLATIONS | BIGINT | num_threshold_violations - Number of threshold violations |

SNAPDBM administrative view and SNAP_GET_DBM table function - Retrieve the dbm logical grouping snapshot information

The SNAPDBM administrative view and the SNAP_GET_DBM table function return the snapshot monitor database manager (dbm) logical grouping information.

Important: The SNAPDBM administrative view and SNAP_GET_DBM table function are deprecated and have been replaced by the “MON_GET_INSTANCE table function - Get instance level information” on page 667.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPDBM administrative view”
- “SNAP_GET_DBM table function” on page 1336

SNAPDBM administrative view

Used with the SNAPSWITCHES administrative view and the MON_GET_FCM, MON_GET_MEMORY_SET, MON_GET_MEMORY_POOL table functions, the SNAPDBM administrative view provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

The schema is SYSIBMADM.

Refer to Table 288 on page 1337 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPDBM administrative view
- CONTROL privilege on the SNAPDBM administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DBM table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve database manager status and connection information for all database members.

```
SELECT DB2_STATUS, DB2START_TIME, LAST_RESET, LOCAL_CONS, REM_CONS_IN,  
       (AGENTS_CREATED_EMPTY_POOL/AGENTS_FROM_POOL) AS AGENT_USAGE,  
       DBPARTITIONNUM FROM SYSIBMADM.SNAPDBM ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| DB2_STATUS | DB2START_TIME | LAST_RESET | ... |
|------------|----------------------------|------------|-------|
| ACTIVE | 2006-01-06-14.59.59.059879 | | - ... |
| ACTIVE | 2006-01-06-14.59.59.097605 | | - ... |
| ACTIVE | 2006-01-06-14.59.59.062798 | | - ... |

3 record(s) selected. ...

Output from this query (continued).

| ... | LOCAL_CONS | REM_CONS_IN | AGENT_USAGE | DBPARTITIONNUM |
|-----|------------|-------------|-------------|----------------|
| ... | 1 | 1 | 0 | 0 |
| ... | 0 | 0 | 0 | 1 |
| ... | 0 | 0 | 0 | 2 |

SNAP_GET_DBM table function

The SNAP_GET_DBM table function returns the same information as the SNAPDBM administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DBM_MEMORY_POOL, SNAP_GET_FCM, SNAP_GET_FCM_PART and SNAP_GET_SWITCHES table functions, the SNAP_GET_DBM table function provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

Refer to Table 288 on page 1337 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_DBM (member) ►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DBM table function calls the snapshot from memory.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DBM table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve the start time and current status of database member number 2.

```
SELECT DB2START_TIME, DB2_STATUS FROM TABLE(SNAP_GET_DBM(2)) AS T
```

The following is an example of output from this query.

```
DB2START_TIME          DB2_STATUS
-----
2006-01-06-14.59.59.062798 ACTIVE
```

Information returned

Table 288. Information returned by the SNAPDBM administrative view and the SNAP_GET_DBM table function

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total sort heap allocated |
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post threshold sorts |
| PIPED_SORTS_REQUESTED | BIGINT | pipedsortsrequested - Piped sorts requested |
| PIPED_SORTS_ACCEPTED | BIGINT | pipedsortsaccepted - Piped sorts accepted |
| REM_CONS_IN | BIGINT | rem_cons_in - Remote connections to database manager |
| REM_CONS_IN_EXEC | BIGINT | rem_cons_in_exec - Remote Connections Executing in the Database Manager monitor element |
| LOCAL_CONS | BIGINT | local_cons - Local connections |
| LOCAL_CONS_IN_EXEC | BIGINT | local_cons_in_exec - Local Connections Executing in the Database Manager monitor element |
| CON_LOCAL_DBASES | BIGINT | con_local_dbases - Local databases with current connects |
| AGENTS_REGISTERED | BIGINT | agents_registered - Agents registered |
| AGENTS_WAITING_ON_TOKEN | BIGINT | agents_waiting_on_token - Agents waiting for a token |
| DB2_STATUS | VARCHAR(12) | db2_status - Status of the database instance

This interface returns a text identifier based on definitions in sqlmon.h, and is one of: <ul style="list-style-type: none">• ACTIVE• QUIESCE_PEND• QUIESCED |
| AGENTS_REGISTERED_TOP | BIGINT | agents_registered_top - Maximum number of agents registered |

Table 288. Information returned by the SNAPDBM administrative view and the SNAP_GET_DBM table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|--|
| AGENTS_WAITING_TOP | BIGINT | agents_waiting_top - Maximum number of agents waiting |
| COMM_PRIVATE_MEM | BIGINT | comm_private_mem - Committed private memory |
| IDLE_AGENTS | BIGINT | idle_agents - Number of idle agents |
| AGENTS_FROM_POOL | BIGINT | agents_from_pool - Agents assigned from pool |
| AGENTS_CREATED_EMPTY_POOL | BIGINT | agents_created_empty_pool - Agents created due to empty agent pool |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum number of coordinating agents |
| MAX_AGENT_OVERFLOW | BIGINT | max_agent_overflows - Maximum agent overflows |
| AGENTS_STOLEN | BIGINT | agents_stolen - Stolen agents |
| GW_TOTAL_CONS | BIGINT | gw_total_cons - Total number of attempted connections for Db2 Connect |
| GW_CUR_CONS | BIGINT | gw_cur_cons - Current number of connections for Db2 Connect |
| GW_CONS_WAIT_HOST | BIGINT | gw_cons_wait_host - Number of connections waiting for the host to reply |
| GW_CONS_WAIT_CLIENT | BIGINT | gw_cons_wait_client - Number of connections waiting for the client to send request |
| POST_THRESHOLD_HASH_JOINS | BIGINT | post_threshold_hash_joins - Hash join threshold |
| NUM_GW_CONN_SWITCHES | BIGINT | num_gw_conn_switches - Connection switches |
| DB2START_TIME | TIMESTAMP | db2start_time - Start database manager timestamp |
| LAST_RESET | TIMESTAMP | last_reset - Last reset timestamp |
| NUM_NODES_IN_DB2_INSTANCE | INTEGER | num_nodes_in_db2_instance - Number of nodes in database partition |
| PRODUCT_NAME | VARCHAR(32) | product_name - Product name |
| SERVICE_LEVEL | VARCHAR(18) | service_level - Service level |
| SORT_HEAP_TOP | BIGINT | sort_heap_top - Sort private heap high water mark |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| POST_THRESHOLD_OLAP_FUNCS | BIGINT | post_threshold_olap_funcs - OLAP function threshold |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPDBM_MEMORY_POOL administrative view and SNAP_GET_DBM_MEMORY_POOL table function - Retrieve database manager level memory usage information

The SNAPDBM_MEMORY_POOL administrative view and the SNAP_GET_DBM_MEMORY_POOL table function return information about memory usage at the database manager.

Important: Starting in Version 9.7 Fix Pack 5, the SNAPDBM_MEMORY_POOL administrative view and SNAP_GET_DBM_MEMORY_POOL table function have been deprecated and replaced by the “MON_GET_MEMORY_POOL - get memory pool information” on page 679 and “MON_GET_MEMORY_SET - get memory set information” on page 682.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPDBM_MEMORY_POOL administrative view”
- “SNAP_GET_DBM_MEMORY_POOL table function” on page 1340

SNAPDBM_MEMORY_POOL administrative view

Used with the SNAPSWITCHES administrative view and the MON_GET_FCM, MON_GET_INSTANCE, MON_GET_MEMORY_SET, and MON_GET_MEMORY_POOL table functions, the SNAPDBM_MEMORY_POOL administrative view provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

The schema is SYSIBMADM.

Refer to Table 289 on page 1341 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- SELECT privilege on the SNAPDBM_MEMORY_POOL administrative view
- CONTROL privilege on the SNAPDBM_MEMORY_POOL administrative view
- DATAACCESS authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DBM_MEMORY_POOL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of the memory pools and their current size for the database manager of the connected database.

```
SELECT POOL_ID, POOL_CUR_SIZE FROM SNAPDBM_MEMORY_POOL
```

The following is an example of output from this query.

| POOL_ID | POOL_CUR_SIZE |
|---------|---------------|
| MONITOR | 65536 |
| OTHER | 29622272 |
| FCMBP | 57606144 |
| ... | |

SNAP_GET_DBM_MEMORY_POOL table function

The SNAP_GET_DBM_MEMORY_POOL table function returns the same information as the SNAPDBM_MEMORY_POOL administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DBM, SNAP_GET_FCM, SNAP_GET_FCM_PART and SNAP_GET_SWITCHES table functions, the SNAP_GET_DBM_MEMORY_POOL table function provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

Refer to Table 289 on page 1341 for a complete list of information that can be returned.

Syntax

►►—SNAP_GET_DBM_MEMORY_POOL—(—member—)—————►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DBM_MEMORY_POOL table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DBM_MEMORY_POOL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of the memory pools and their current size for all database partitions of the database manager of the connected database.

```
SELECT POOL_ID, POOL_CUR_SIZE, DBPARTITIONNUM
FROM TABLE(SYSPROC.SNAP_GET_DBM_MEMORY_POOL())
AS T ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| POOL_ID | POOL_CUR_SIZE | DBPARTITIONNUM |
|---------|---------------|----------------|
| MONITOR | 65536 | 0 |
| OTHER | 29622272 | 0 |
| FCMBP | 57606144 | 0 |
| MONITOR | 65536 | 1 |
| OTHER | 29425664 | 1 |
| FCMBP | 57606144 | 1 |
| MONITOR | 65536 | 2 |
| OTHER | 29425664 | 2 |
| FCMBP | 57606144 | 2 |

Information returned

Table 289. Information returned by the SNAPDBM_MEMORY_POOL administrative view and the SNAP_GET_DBM_MEMORY_POOL table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |

Table 289. Information returned by the SNAPDBM_MEMORY_POOL administrative view and the SNAP_GET_DBM_MEMORY_POOL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------|-------------|---|
| POOL_ID | VARCHAR(14) | pool_id - Memory pool identifier. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • APP_GROUP • APPL_CONTROL • APPLICATION • BP • CAT_CACHE • DATABASE • DFM • FCMBP • IMPORT_POOL • LOCK_MGR • MONITOR • OTHER • PACKAGE_CACHE • QUERY • SHARED_SORT • SORT • STATEMENT • STATISTICS • UTILITY |
| POOL_CUR_SIZE | BIGINT | pool_cur_size - Current size of memory pool |
| POOL_WATERMARK | BIGINT | pool_watermark - Memory pool watermark |
| POOL_CONFIG_SIZE | BIGINT | pool_config_size - Configured size of memory pool |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAP_GET_DBM_V95 table function - Retrieve the dbm logical grouping snapshot information

The SNAP_GET_DBM_V95 table function returns the snapshot monitor database manager (dbm) logical grouping information.

Important: The SNAP_GET_DBM_V95 table function is deprecated and have been replaced by the “MON_GET_INSTANCE table function - Get instance level information” on page 667.

Refer to Table 290 on page 1344 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_DBM_V95 (member) ►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DBM_V95 table function calls the snapshot from memory.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DBM_V95 table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve the start time and current status of database member number 2.

```
SELECT DB2START_TIME, DB2_STATUS FROM TABLE(SNAP_GET_DBM_V95(2)) AS T
```

The following is an example of output from this query.

| DB2START_TIME | DB2_STATUS |
|----------------------------|------------|
| ----- | |
| 2006-01-06-14.59.59.062798 | ACTIVE |

Information returned

Table 290. Information returned by SNAP_GET_DBM_V95 table function

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total sort heap allocated |
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post threshold sorts |
| PIPED_SORTS_REQUESTED | BIGINT | pipedsortsrequested - Piped sorts requested |
| PIPED_SORTS_ACCEPTED | BIGINT | pipedsortsaccepted - Piped sorts accepted |
| REM_CONS_IN | BIGINT | rem_cons_in - Remote connections to database manager |
| REM_CONS_IN_EXEC | BIGINT | rem_cons_in_exec - Remote Connections Executing in the Database Manager monitor element |
| LOCAL_CONS | BIGINT | local_cons - Local connections |
| LOCAL_CONS_IN_EXEC | BIGINT | local_cons_in_exec - Local Connections Executing in the Database Manager monitor element |
| CON_LOCAL_DBASES | BIGINT | con_local_dbases - Local databases with current connects |
| AGENTS_REGISTERED | BIGINT | agents_registered - Agents registered |
| AGENTS_WAITING_ON_TOKEN | BIGINT | agents_waiting_on_token - Agents waiting for a token |
| DB2_STATUS | VARCHAR(12) | db2_status - Status of database instance

This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • ACTIVE • QUIESCE_PEND • QUIESCED |
| AGENTS_REGISTERED_TOP | BIGINT | agents_registered_top - Maximum number of agents registered |
| AGENTS_WAITING_TOP | BIGINT | agents_waiting_top - Maximum number of agents waiting |
| COMM_PRIVATE_MEM | BIGINT | comm_private_mem - Committed private memory |
| IDLE_AGENTS | BIGINT | idle_agents - Number of idle agents |
| AGENTS_FROM_POOL | BIGINT | agents_from_pool - Agents assigned from pool |
| AGENTS_CREATED_EMPTY_POOL | BIGINT | agents_created_empty_pool - Agents created due to empty agent pool |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum number of coordinating agents |
| MAX_AGENT_OVERFLOW | BIGINT | max_agent_overflows - Maximum agent overflows |
| AGENTS_STOLEN | BIGINT | agents_stolen - Stolen agents |

Table 290. Information returned by SNAP_GET_DBM_V95 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|--|
| GW_TOTAL_CONS | BIGINT | gw_total_cons - Total number of attempted connections for Db2 Connect |
| GW_CUR_CONS | BIGINT | gw_cur_cons - Current number of connections for Db2 Connect |
| GW_CONS_WAIT_HOST | BIGINT | gw_cons_wait_host - Number of connections waiting for the host to reply |
| GW_CONS_WAIT_CLIENT | BIGINT | gw_cons_wait_client - Number of connections waiting for the client to send request |
| POST_THRESHOLD_HASH_JOINS | BIGINT | post_threshold_hash_joins - Hash join threshold |
| NUM_GW_CONN_SWITCHES | BIGINT | num_gw_conn_switches - Connection switches |
| DB2START_TIME | TIMESTAMP | db2start_time - Start database manager timestamp |
| LAST_RESET | TIMESTAMP | last_reset - Last reset timestamp |
| NUM_NODES_IN_DB2_INSTANCE | INTEGER | num_nodes_in_db2_instance - Number of nodes in database partition |
| PRODUCT_NAME | VARCHAR(32) | product_name - Product name |
| SERVICE_LEVEL | VARCHAR(18) | service_level - Service level |
| SORT_HEAP_TOP | BIGINT | sort_heap_top - Sort private heap high water mark |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| POST_THRESHOLD_OLAP_FUNCS | BIGINT | post_threshold_olap_funcs - OLAP function threshold |

SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG table function - Retrieve snapshot information from the detail_log logical data group

The SNAPDETAILLOG administrative view and the SNAP_GET_DETAILLOG table function return snapshot information from the detail_log logical data group.

Important: The SNAPDETAILLOG administrative view and the SNAP_GET_DETAILLOG table function are deprecated and have been replaced by the “MON_GET_TRANSACTION_LOG table function - Get log information” on page 827.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPDETAILLOG administrative view” on page 1346
- “SNAP_GET_DETAILLOG table function” on page 1347

SNAPDETAILLOG administrative view

This administrative view allows you to retrieve snapshot information from the detail_log logical data group for the currently connected database.

Used in conjunction with ADMIN_GET_STORAGE_PATHS, MON_GET_DATABASE, MON_GET_HADR, MON_GET_MEMORY_POOL, and MON_GET_MEMORY_SET table functions, the SNAPDETAILLOG administrative view provides information equivalent to the **GET SNAPSHOT FOR DATABASE on database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 291 on page 1348 for a complete list of information that is returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPDETAILLOG administrative view
- CONTROL privilege on the SNAPDETAILLOG administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DETAILLOG table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve log information for all database members for the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 8) AS DB_NAME, FIRST_ACTIVE_LOG,  
       LAST_ACTIVE_LOG, CURRENT_ACTIVE_LOG, CURRENT_ARCHIVE_LOG,  
       DBPARTITIONNUM  
FROM SYSIBMADM.SNAPDETAILLOG ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| DB_NAME | FIRST_ACTIVE_LOG | LAST_ACTIVE_LOG | ... |
|---------|------------------|-----------------|-----|
| TEST | 0 | 8 | ... |
| TEST | 0 | 8 | ... |
| TEST | 0 | 8 | ... |

3 record(s) selected.

Output from this query (continued).

| ... | CURRENT_ACTIVE_LOG | CURRENT_ARCHIVE_LOG | DBPARTITIONNUM |
|-----|--------------------|---------------------|----------------|
| ... | 0 | - | 0 |
| ... | 0 | - | 1 |
| ... | 0 | - | 2 |

SNAP_GET_DETAILLOG table function

The SNAP_GET_DETAILLOG table function returns the same information as the SNAPDETAILLOG administrative view.

Used in conjunction with ADMIN_GET_STORAGE_PATHS, MON_GET_HADR, MON_GET_MEMORY_POOL, and SNAP_GET_DB, the SNAPDETAILLOG administrative view provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES** CLP command.

Refer to Table 291 on page 1348 for a complete list of information that is returned.

Syntax

```

-- SNAP_GET_DETAILLOG ( ( dbname [ , member ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the

SNAP_GET_DETAILLOG table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DETAILLOG table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve log information for database member 1 for the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 8) AS DB_NAME, FIRST_ACTIVE_LOG,
       LAST_ACTIVE_LOG, CURRENT_ACTIVE_LOG, CURRENT_ARCHIVE_LOG
FROM TABLE(SNAP_GET_DETAILLOG(' ', 1)) AS T
```

The following is an example of output from this query.

```
DB_NAME  FIRST_ACTIVE_LOG  LAST_ACTIVE_LOG  ...
-----
TEST          0          8 ...
```

1 record(s) selected.

Output from this query (continued).

```
... CURRENT_ACTIVE_LOG  CURRENT_ARCHIVE_LOG
... -----
...          0          -
```

Information returned

Table 291. Information returned by the SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| FIRST_ACTIVE_LOG | BIGINT | first_active_log - First active log file number |
| LAST_ACTIVE_LOG | BIGINT | last_active_log - Last active log file number |

Table 291. Information returned by the SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|-----------|--|
| CURRENT_ACTIVE_LOG | BIGINT | current_active_log - Current active log file number |
| CURRENT_ARCHIVE_LOG | BIGINT | current_archive_log - Current archive log file number |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAP_GET_DETAILLOG_V91 table function - Retrieve snapshot information from the detail_log logical data group

The SNAP_GET_DETAILLOG_V91 table function returns snapshot information from the detail_log logical data group.

Note: This table function has been deprecated and replaced by the “MON_GET_TRANSACTION_LOG table function - Get log information” on page 827.

Refer to Table 292 on page 1350 for a complete list of information that is returned.

Syntax

```

▶▶ SNAP_GET_DETAILLOG_V91 ( ( dbname [ , dbpartitionnum ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

dbpartitionnum

An optional input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for an aggregate of all active database partitions. If *dbname* is not set to NULL and *dbpartitionnum* is set to NULL, -1 is set implicitly for *dbpartitionnum*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database partitions. An active database partition is a partition where the database is available for connection and use by applications.

If both *dbname* and *dbpartitionnum* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could

have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DETAILLOG_V91 table function takes a snapshot for the currently connected database and database partition number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DETAILLOG_V91 table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve log information for database partition 1 for the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 8) AS DB_NAME, FIRST_ACTIVE_LOG,
       LAST_ACTIVE_LOG, CURRENT_ACTIVE_LOG, CURRENT_ARCHIVE_LOG
FROM TABLE(SNAP_GET_DETAILLOG_V91(' ', 1)) AS T
```

The following is an example of output from this query.

```
DB_NAME  FIRST_ACTIVE_LOG  LAST_ACTIVE_LOG  ...
-----
TEST          0          8 ...
```

1 record(s) selected.

Output from this query (continued).

```
... CURRENT_ACTIVE_LOG  CURRENT_ARCHIVE_LOG
... -----
...          0          -
```

Information returned

Table 292. Information returned by the SNAP_GET_DETAILLOG_V91 table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| FIRST_ACTIVE_LOG | BIGINT | first_active_log - First active log file number |

Table 292. Information returned by the SNAP_GET_DETAILLOG_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|-----------|--|
| LAST_ACTIVE_LOG | BIGINT | last_active_log - Last active log file number |
| CURRENT_ACTIVE_LOG | BIGINT | current_active_log - Current active log file number |
| CURRENT_ARCHIVE_LOG | BIGINT | current_archive_log - Current archive log file number |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

SNAP_GET_DYN_SQL_V95 table function - Retrieve dynsql logical group snapshot information

The SNAP_GET_DYN_SQL_V95 table function returns snapshot information from the dynsql logical data group.

Note: This table function has been deprecated and replaced by the "SNAP_GET_DYN_SQL table function" on page 942.

Refer to Table 293 on page 1353 for a complete list of information that can be returned.

Syntax

```

▶▶ SNAP_GET_DYN_SQL_V95 (—dbname—, member—)

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the

SNAP_GET_DYN_SQL_V95 table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DYN_SQL_V95 table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of dynamic SQL run on the currently connected database member of the currently connected database, ordered by the number of rows read.

```
SELECT PREP_TIME_WORST, NUM_COMPILATIONS, SUBSTR(STMT_TEXT, 1, 60)
      AS STMT_TEXT FROM TABLE(SNAP_GET_DYN_SQL_V95('','-1)) as T
      ORDER BY ROWS_READ
```

The following is an example of output from this query.

| PREP_TIME_WORST | ... |
|-----------------|-----|
| ----- | ... |
| 0 | ... |
| 3 | ... |
| ... | ... |
| 4 | ... |
| ... | ... |
| 4 | ... |
| ... | ... |
| 4 | ... |
| ... | ... |
| 3 | ... |
| ... | ... |
| 4 | ... |
| ... | ... |

Output from this query (continued).

| NUM_COMPILATIONS | STMT_TEXT |
|------------------|---------------------------------------|
| ----- | ----- |
| 0 | SET CURRENT LOCALE LC_CTYPE = 'en_US' |
| 1 | select rows_read, rows_written, |
| | substr(stmt_text, 1, 40) as |
| 1 | select * from table |
| | (snap_get_dyn_sqlv9('','-1)) as t |
| 1 | select * from table |
| | (snap_getdetaillog9('','-1)) as t |
| 1 | select * from table |
| | (snap_get_hadr('','-1)) as t |

```

...      1 select prep_time_worst, num_compilations,
...          substr(stmt_text,
...      1 select prep_time_worst, num_compilations,
...          substr(stmt_text,

```

After running a workload, user can use the following query with the table function.

```

SELECT STATS_FABRICATE_TIME,SYNC_RUNSTATS_TIME
FROM TABLE (SNAP_GET_DYN_SQL_V95('mytestdb', -1))
AS SNAPDB

```

```

STATS_FABRICATE_TIME  SYNC_RUNSTATS_TIME
-----
                      2                12
                      1                30

```

For the view based on this table function:

```

SELECT STATS_FABRICATE_TIME,SYNC_RUNSTATS_TIME
FROM SYSIBMADM.SNAPDYN_SQL

```

```

STATS_FABRICATE_TIME  SYNC_RUNSTATS_TIME
-----
                      5                10
                      3                20

```

2 record(s) selected.

Information returned

Table 293. Information returned by the SNAP_GET_DYN_SQL_V95 table function

| Column name | Data type | Description or corresponding monitor element |
|------------------------|-----------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| NUM_EXECUTIONS | BIGINT | num_executions - Statement executions |
| NUM_COMPILATIONS | BIGINT | num_compilations - Statement compilations |
| PREP_TIME_WORST | BIGINT | prep_time_worst - Statement worst preparation time |
| PREP_TIME_BEST | BIGINT | prep_time_best - Statement best preparation time |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal rows deleted |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal rows inserted |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal rows updated |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| STMT_SORTS | BIGINT | stmt_sorts - Statement sorts |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total sort time |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |

Table 293. Information returned by the SNAP_GET_DYN_SQL_V95 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--|-----------|--|
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| TOTAL_EXEC_TIME | BIGINT | total_exec_time - Elapsed statement execution time (in seconds)* |
| TOTAL_EXEC_TIME_MS | BIGINT | total_exec_time - Elapsed statement execution time (fractional, in microseconds)* |
| TOTAL_USR_CPU_TIME | BIGINT | total_usr_cpu_time - Total user CPU for a statement (in seconds)* |
| TOTAL_USR_CPU_TIME_MS | BIGINT | total_usr_cpu_time - Total user CPU for a statement (fractional, in microseconds)* |
| TOTAL_SYS_CPU_TIME | BIGINT | total_sys_cpu_time - Total system CPU for a statement (in seconds)* |
| TOTAL_SYS_CPU_TIME_MS | BIGINT | total_sys_cpu_time - Total system CPU for a statement (fractional, in microseconds)* |
| STMT_TEXT | CLOB(2 M) | stmt_text - SQL statement text |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| STATS_FABRICATE_TIME | BIGINT | The total time (in milliseconds) spent by system to create needed statistics without table or index scan during query compilation for a dynamic statement. |
| SYNC_RUNSTATS_TIME | BIGINT | The total time (in milliseconds) spent on synchronous statistics-collect activities during query compilation for a dynamic statement. |
| <p>* To calculate the total time spent for the monitor element that this column is based on, you must add the full seconds reported in the column for this monitor element that ends with _S to the fractional seconds reported in the column for this monitor element that ends with _MS, using the following formula: $(\text{monitor-element-name_S} \times 1,000,000 + \text{monitor-element-name_MS}) \div 1,000,000$. For example, $(\text{ELAPSED_EXEC_TIME_S} \times 1,000,000 + \text{ELAPSED_EXEC_TIME_MS}) \div 1,000,000$.</p> | | |

SNAPFCM administrative view and SNAP_GET_FCM table function - Retrieve the fcm logical data group snapshot information

The SNAPFCM administrative view and the SNAP_GET_FCM table function return information about the fast communication manager from a database manager snapshot, in particular, the fcm logical data group.

Important: The SNAPFCM administrative view and the SNAP_GET_FCM table function are deprecated and have been replaced by the “MON_GET_FCM - Get FCM metrics” on page 652.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPFCM administrative view”
- “SNAP_GET_FCM table function” on page 1356

SNAPFCM administrative view

Used with the SNAPSWITCHES administrative view and the MON_GET_FCM, MON_GET_INSTANCE, MON_GET_MEMORY_SET, and MON_GET_MEMORY_POOL table functions, the SNAPFCM administrative view provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

The schema is SYSIBMADM.

Refer to Table 294 on page 1357 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPFCM administrative view
- CONTROL privilege on the SNAPFCM administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_FCM table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve information about the fast communication manager's message buffers on all members.

```
SELECT BUFF_FREE, BUFF_FREE_BOTTOM, MEMBER
FROM SYSIBMADM.SNAPFCM ORDER BY MEMBER
```

The following is an example of output from this query.

| BUFF_FREE | BUFF_FREE_BOTTOM | MEMBER |
|-----------|------------------|--------|
| 5120 | 5100 | 0 |
| 5120 | 5100 | 1 |
| 5120 | 5100 | 2 |

SNAP_GET_FCM table function

The SNAP_GET_FCM table function returns the same information as the SNAPFCM administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DBM, SNAP_GET_DBM_MEMORY_POOL, SNAP_GET_FCM_PART and SNAP_GET_SWITCHES table functions, the SNAP_GET_FCM table function provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

Refer to Table 294 on page 1357 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_FCM (member) ►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current member, or -2 for an aggregate of all active members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_FCM

table function takes a snapshot for the currently connected database and database member.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_FCM table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve information about the fast communication manager's message buffers on database member 1.

```
SELECT BUFF_FREE, BUFF_FREE_BOTTOM, MEMBER
FROM TABLE(SYSPROC.SNAP_GET_FCM( 1 )) AS T
```

The following is an example of output from this query.

| BUFF_FREE | BUFF_FREE_BOTTOM | MEMBER |
|-----------|------------------|--------|
| 5120 | 5100 | 1 |

Information returned

Table 294. Information returned by the SNAPFCM administrative view and the SNAP_GET_FCM table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| BUFF_FREE | BIGINT | buff_free - FCM buffers currently free |
| BUFF_FREE_BOTTOM | BIGINT | buff_free_bottom - Minimum FCM Buffers Free |
| CH_FREE | BIGINT | ch_free - Channels Currently Free |
| CH_FREE_BOTTOM | BIGINT | ch_free_bottom - Minimum Channels Free |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function - Retrieve the fcm_node logical data group snapshot information

The SNAPFCM_PART administrative view and the SNAP_GET_FCM_PART table function return information about the fast communication manager from a database manager snapshot, in particular, the fcm_node logical data group.

Important: The SNAPFCM_PART administrative view and the SNAP_GET_FCM_PART table function are deprecated and have been replaced by the “MON_GET_FCM - Get FCM metrics” on page 652.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPFCM_PART administrative view”
- “SNAP_GET_FCM_PART table function” on page 1359

SNAPFCM_PART administrative view

Used with the SNAPSWITCHES administrative view and the MON_GET_FCM, MON_GET_INSTANCE, MON_GET_MEMORY_SET, and MON_GET_MEMORY_POOL table functions, the SNAPFCM_PART administrative view provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

The schema is SYSIBMADM.

Refer to Table 295 on page 1360 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPFCM_PART administrative view
- CONTROL privilege on the SNAPFCM_PART administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_FCM_PART table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve buffers sent and received information for the fast communication manager.

```
SELECT CONNECTION_STATUS, TOTAL_BUFFERS_SENT, TOTAL_BUFFERS_RECEIVED
FROM SYSIBADM.SNAPFCM_PART WHERE MEMBER = 0
```

The following is an example of output from this query.

| CONNECTION_STATUS | TOTAL_BUFFERS_SENT | TOTAL_BUFFERS_RCVD |
|-------------------|--------------------|--------------------|
| INACTIVE | 2 | 1 |

1 record(s) selected.

SNAP_GET_FCM_PART table function

The SNAP_GET_FCM_PART table function returns the same information as the SNAPFCM_PART administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DBM, SNAP_GET_DBM_MEMORY_POOL, SNAP_GET_FCM and SNAP_GET_SWITCHES table functions, the SNAP_GET_FCM_PART table function provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

Refer to Table 295 on page 1360 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_FCM_PART (member) ►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current member. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_FCM_PART table function takes a snapshot for the currently connected database and member.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_FCM_PART table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve buffers sent and received information for the fast communication manager for all members.

```
SELECT FCM_MEMBER, TOTAL_BUFFERS_SENT, TOTAL_BUFFERS_RCVD,  
       MEMBER FROM TABLE(SNAP_GET_FCM_PART()) AS T  
ORDER BY MEMBER
```

The following is an example of output from this query.

| FCM_MEMBER | TOTAL_BUFFERS_SENT | TOTAL_BUFFERS_RCVD | MEMBER |
|------------|--------------------|--------------------|--------|
| 0 | 305 | 305 | 0 |
| 1 | 5647 | 1664 | 0 |
| 2 | 5661 | 1688 | 0 |
| 0 | 19 | 19 | 1 |
| 1 | 305 | 301 | 1 |
| 2 | 1688 | 5661 | 1 |
| 0 | 1664 | 5647 | 2 |
| 1 | 10 | 10 | 2 |
| 2 | 301 | 305 | 2 |

Information returned

Table 295. Information returned by the SNAPFCM_PART administrative view and the SNAP_GET_FCM_PART table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| CONNECTION_STATUS | VARCHAR(10) | connection_status - Connection status. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none">• INACTIVE• ACTIVE• CONGESTED |

Table 295. Information returned by the SNAPFCM_PART administrative view and the SNAP_GET_FCM_PART table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| TOTAL_BUFFERS_SENT | BIGINT | total_buffers_sent - Total FCM buffers sent |
| TOTAL_BUFFERS_RCVD | BIGINT | total_buffers_rcvd - Total FCM buffers received |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| FCM_DBPARTITIONNUM | SMALLINT | The database partition number to which data was sent or from which data was received (as per the TOTAL_BUFFERS_SENT and TOTAL_BUFFERS_RCVD columns). |
| MEMBER | SMALLINT | member - Database member monitor element |
| FCM_MEMBER | SMALLINT | The member to which data was sent or from which data was received (as per the TOTAL_BUFFERS_SENT and TOTAL_BUFFERS_RCVD columns). |

SNAPHADR administrative view and SNAP_GET_HADR table function - Retrieve hadr logical data group snapshot information

The SNAPHADR administrative view and the SNAP_GET_HADR table function return information about high availability disaster recovery from a database snapshot, in particular, the hadr logical data group.

The SNAPHADR administrative view and SNAP_GET_HADR table function have been deprecated and replaced by the “MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information” on page 656.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPHADR administrative view”
- “SNAP_GET_HADR table function” on page 1362

SNAPHADR administrative view

This administrative view allows you to retrieve hadr logical data group snapshot information for the currently connected database. The data is only returned by this view if the database is a primary or standby high availability disaster recovery (HADR) database.

Used with the MON_GET_DATABASE, MON_GET_MEMORY_SET, MON_GET_MEMORY_POOL, MON_GET_TRANSACTION_LOG and

ADMIN_GET_STORAGE_PATHS table functions, the SNAPHADR administrative view provides information equivalent to the **GET SNAPSHOT FOR DATABASE ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 296 on page 1364 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- SELECT privilege on the SNAPHADR administrative view
- CONTROL privilege on the SNAPHADR administrative view
- DATAACCESS authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_HADR table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve the configuration and status information for HADR on the primary HADR database.

```
SELECT SUBSTR(DB_NAME, 1, 8) AS DBNAME, HADR_ROLE, HADR_STATE,
       HADR_SYNCMODE, HADR_CONNECT_STATUS
FROM SYSIBMADM.SNAPHADR
```

The following is an example of output from this query.

| DBNAME | HADR_ROLE | HADR_STATE | HADR_SYNCMODE | HADR_CONNECT_STATUS |
|--------|-----------|------------|---------------|---------------------|
| ----- | ----- | ----- | ----- | ----- |
| SAMPLE | PRIMARY | PEER | SYNC | CONNECTED |

1 record(s) selected.

SNAP_GET_HADR table function

The SNAP_GET_HADR table function returns the same information as the SNAPHADR administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DB, SNAP_GET_DB_MEMORY_POOL, SNAP_GET_DETAILLOG and ADMIN_GET_STORAGE_PATHS table functions, the SNAP_GET_HADR table function provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES** CLP command.

Refer to Table 296 on page 1364 for a complete list of information that can be returned.

Syntax

```

▶▶ SNAP_GET_HADR ( ( dbname [ , member ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_HADR table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_HADR table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve the configuration and status information for HADR for all databases.

```
SELECT SUBSTR(DB_NAME, 1, 8) AS DBNAME, HADR_ROLE, HADR_STATE,  
       HADR_SYNCMODE, HADR_CONNECT_STATUS  
FROM TABLE (SNAP_GET_HADR (CAST (NULL as VARCHAR(128)), 0)) as T
```

The following is an example of output from this query.

| DBNAME | HADR_ROLE | HADR_STATE | HADR_SYNCMODE | HADR_CONNECT_STATUS |
|--------|-----------|--------------|---------------|---------------------|
| SAMPLE | PRIMARY | PEER | SYNC | CONNECTED |
| TESTDB | PRIMARY | DISCONNECTED | NEARSYNC | DISCONNECTED |

2 record(s) selected.

Information returned

Table 296. Information returned by the SNAPHADR administrative view and the SNAP_GET_HADR table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| HADR_ROLE | VARCHAR(10) | hadr_role - HADR role. This interface returns a text identifier based on the defines in sqlmon.h, and is one of: <ul style="list-style-type: none">• PRIMARY• STANDARD• STANDBY |
| HADR_STATE | VARCHAR(14) | hadr_state - HADR state. This interface returns a text identifier based on the defines in sqlmon.h, and is one of: <ul style="list-style-type: none">• DISCONNECTED• LOCAL_CATCHUP• PEER• REM_CATCH_PEN• REM_CATCHUP |
| HADR_SYNCMODE | VARCHAR(10) | hadr_syncmode - HADR synchronization mode. This interface returns a text identifier based on the defines in sqlmon.h, and is one of: <ul style="list-style-type: none">• ASYNC• NEARSYNC• SUPERASYNC• SYNC |

Table 296. Information returned by the SNAPHADR administrative view and the SNAP_GET_HADR table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|--------------|--|
| HADR_CONNECT_STATUS | VARCHAR(12) | hadr_connect_status - HADR connection status. This interface returns a text identifier based on the defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • CONGESTED • CONNECTED • DISCONNECTED |
| HADR_CONNECT_TIME | TIMESTAMP | hadr_connect_time - HADR connection time |
| HADR_HEARTBEAT | INTEGER | hadr_heartbeat - HADR heartbeat |
| HADR_LOCAL_HOST | VARCHAR(255) | hadr_local_host - HADR local host |
| HADR_LOCAL_SERVICE | VARCHAR(40) | hadr_local_service - HADR local service |
| HADR_REMOTE_HOST | VARCHAR(255) | hadr_remote_host - HADR remote host |
| HADR_REMOTE_SERVICE | VARCHAR(40) | hadr_remote_service - HADR remote service |
| HADR_REMOTE_INSTANCE | VARCHAR(128) | hadr_remote_instance - HADR remote instance |
| HADR_TIMEOUT | BIGINT | hadr_timeout - HADR timeout |
| HADR_PRIMARY_LOG_FILE | VARCHAR(255) | hadr_primary_log_file - HADR primary log file |
| HADR_PRIMARY_LOG_PAGE | BIGINT | hadr_primary_log_page - HADR primary log page |
| HADR_PRIMARY_LOG_LSN | BIGINT | hadr_primary_log_lsn - HADR primary log LSN |
| HADR_STANDBY_LOG_FILE | VARCHAR(255) | hadr_standby_log_file - HADR standby log file |
| HADR_STANDBY_LOG_PAGE | BIGINT | hadr_standby_log_page - HADR standby log page |
| HADR_STANDBY_LOG_LSN | BIGINT | hadr_standby_log_lsn - HADR standby log LSN |
| HADR_LOG_GAP | BIGINT | hadr_log_gap - HADR log gap |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPLOCK administrative view and SNAP_GET_LOCK table function - Retrieve lock logical data group snapshot information

The SNAPLOCK administrative view and the SNAP_GET_LOCK table function return snapshot information about locks, in particular, the lock logical data group.

Note: This administrative view and table function have been deprecated and replaced by the “MON_GET_APPL_LOCKWAIT - Get information about locks for which an application is waiting” on page 561, “MON_GET_LOCKS - List all locks in the currently connected database” on page 674, and “MON_FORMAT_LOCK_NAME - Format the internal lock name and return details” on page 483.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPLOCK administrative view”
- “SNAP_GET_LOCK table function” on page 1367

SNAPLOCK administrative view

This administrative view allows you to retrieve lock logical data group snapshot information for the currently connected database.

Used with the SNAPLOCKWAIT administrative view, the SNAPLOCK administrative view provides information equivalent to the **GET SNAPSHOT FOR LOCKS ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 297 on page 1368 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- SELECT privilege on the SNAPLOCK administrative view
- CONTROL privilege on the SNAPLOCK administrative view
- DATAACCESS authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_LOCK table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve lock information for the database member 0 of the currently connected database.

```
SELECT AGENT_ID, LOCK_OBJECT_TYPE, LOCK_MODE, LOCK_STATUS
FROM SYSIBMADM.SNAPLOCK WHERE DBPARTITIONNUM = 0
```

The following is an example of output from this query.

| AGENT_ID | LOCK_OBJECT_TYPE | LOCK_MODE | LOCK_STATUS |
|----------|------------------|-----------|-------------|
| 7 | TABLE | IX | GRNT |

1 record(s) selected.

SNAP_GET_LOCK table function

The SNAP_GET_LOCK table function returns the same information as the SNAPLOCK administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_LOCKWAIT table function, the SNAP_GET_LOCK table function provides information equivalent to the **GET SNAPSHOT FOR LOCKS ON database-alias** CLP command.

Refer to Table 297 on page 1368 for a complete list of information that can be returned.

Syntax

```
➤ SNAP_GET_LOCK ( ( dbname [ , member ] ) ) ➤
```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify a null value or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_LOCK table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_LOCK table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve lock information for the current database member of the currently connected database.

```
SELECT AGENT_ID, LOCK_OBJECT_TYPE, LOCK_MODE, LOCK_STATUS
FROM TABLE(SNAP_GET_LOCK('I',-1)) as T
```

The following is an example of output from this query.

| AGENT_ID | LOCK_OBJECT_TYPE | LOCK_MODE | LOCK_STATUS |
|----------|------------------|-----------|-------------|
| 680 | INTERNALV_LOCK | S | GRNT |
| 680 | INTERNALP_LOCK | S | GRNT |

2 record(s) selected.

Information returned

Table 297. Information returned by the SNAPLOCK administrative view and the SNAP_GET_LOCK table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| TAB_FILE_ID | BIGINT | table_file_id - Table file identification |

Table 297. Information returned by the SNAPLOCK administrative view and the SNAP_GET_LOCK table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------|-------------|--|
| LOCK_OBJECT_TYPE | VARCHAR(18) | lock_object_type - Lock object type waited on. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • AUTORESIZE_LOCK • AUTOSTORAGE_LOCK • BLOCK_LOCK • EOT_LOCK • INPLACE_REORG_LOCK • INTERNAL_LOCK • INTERNALB_LOCK • INTERNALC_LOCK • INTERNALJ_LOCK • INTERNALL_LOCK • INTERNALO_LOCK • INTERNALQ_LOCK • INTERNALP_LOCK • INTERNALS_LOCK • INTERNALT_LOCK • INTERNALV_LOCK • KEYVALUE_LOCK • ROW_LOCK • SYSBOOT_LOCK • TABLE_LOCK • TABLE_PART_LOCK • TABLESPACE_LOCK • XML_PATH_LOCK • COL_TABLE_SERIALIZE_LOCK |
| LOCK_MODE | VARCHAR(10) | lock_mode - Lock mode. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |

Table 297. Information returned by the SNAPLOCK administrative view and the SNAP_GET_LOCK table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------|--------------|---|
| LOCK_STATUS | VARCHAR(10) | lock_status - Lock status. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • CONV • GRNT |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock escalation |
| TABNAME | VARCHAR(128) | table_name - Table name |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| LOCK_ATTRIBUTES | VARCHAR(128) | lock_attributes - Lock attributes. This interface returns a text identifier based on the defines in sqlmon.h. If there are no locks, the text identifier is NONE, otherwise, it is any combination of the following separated by a '+' sign: <ul style="list-style-type: none"> • ALLOW_NEW • DELETE_IN_BLOCK • ESCALATED • INSERT • NEW_REQUEST • RR • RR_IN_BLOCK • UPDATE_DELETE • WAIT_FOR_AVAIL |
| LOCK_COUNT | BIGINT | lock_count - Lock count |
| LOCK_CURRENT_MODE | VARCHAR(10) | lock_current_mode - Original lock mode before conversion. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |
| LOCK_HOLD_COUNT | BIGINT | lock_hold_count - Lock hold count |

Table 297. Information returned by the SNAPLOCK administrative view and the SNAP_GET_LOCK table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|---|
| LOCK_NAME | VARCHAR(32) | lock_name - Lock name |
| LOCK_RELEASE_FLAGS | BIGINT | lock_release_flags - Lock release flags |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition identifier. For a non-partitioned table, this element is NULL. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function - Retrieve lockwait logical data group snapshot information

The SNAPLOCKWAIT administrative view and the SNAP_GET_LOCKWAIT table function return snapshot information about lock waits, in particular, the lockwait logical data group.

Note: This administrative view and table function have been deprecated and replaced by the “MON_LOCKWAITS administrative view - Retrieve metrics for applications that are waiting to obtain locks” on page 441 and the “MON_GET_APPL_LOCKWAIT - Get information about locks for which an application is waiting” on page 561, “MON_GET_LOCKS - List all locks in the currently connected database” on page 674, and “MON_FORMAT_LOCK_NAME - Format the internal lock name and return details” on page 483.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPLOCKWAIT administrative view”
- “SNAP_GET_LOCKWAIT table function” on page 1372

SNAPLOCKWAIT administrative view

This administrative view allows you to retrieve lockwait logical data group snapshot information for the currently connected database.

Used with the SNAPLOCK administrative view, the SNAPLOCKWAIT administrative view provides information equivalent to the **GET SNAPSHOT FOR LOCKS ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 298 on page 1374 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- SELECT privilege on the SNAPLOCKWAIT administrative view
- CONTROL privilege on the SNAPLOCKWAIT administrative view
- DATAACCESS authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_LOCKWAIT table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve lock wait information for database member 0 for the currently connected database.

```
SELECT AGENT_ID, LOCK_MODE, LOCK_OBJECT_TYPE, AGENT_ID_HOLDING_LK,  
       LOCK_MODE_REQUESTED FROM SYSIBMADM.SNAPLOCKWAIT  
WHERE DBPARTITIONNUM = 0
```

The following is an example of output from this query.

| AGENT_ID | LOCK_MODE | LOCK_OBJECT_TYPE | ... |
|----------|-----------|------------------|-----|
| 7 | IX | TABLE | ... |

1 record(s) selected.

Output from this query (continued).

| ... | AGENT_ID_HOLDING_LK | LOCK_MODE_REQUESTED | ... |
|-----|---------------------|---------------------|-----|
| ... | 12 | IS | ... |

SNAP_GET_LOCKWAIT table function

The SNAP_GET_LOCKWAIT table function returns the same information as the SNAPLOCKWAIT administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_LOCK table function, the SNAP_GET_LOCKWAIT table function provides information equivalent to the **GET SNAPSHOT FOR LOCKS ON database-alias** CLP command.

Refer to Table 298 on page 1374 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_LOCKWAIT (—*dbname* — [, *member*]) —►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify a null value or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_LOCKWAIT table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_LOCKWAIT table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve lock wait information for current database member for the currently connected database.

```
SELECT AGENT_ID, LOCK_MODE, LOCK_OBJECT_TYPE, AGENT_ID_HOLDING_LK,  
       LOCK_MODE_REQUESTED FROM TABLE(SNAP_GET_LOCKWAIT('','-1)) AS T
```

The following is an example of output from this query.

```
AGENT_ID      LOCK_MODE  LOCK_OBJECT_TYPE  ...  
-----  
          12 X          ROW_LOCK      ...
```

1 record(s) selected.

Output from this query (continued).

```
... AGENT_ID_HOLDING_LK  LOCK_MODE_REQUESTED  
... -----  
...                   7 X
```

Usage note

To see lock wait information, you must first turn on the default LOCK monitor switch in the database manager configuration. To have the change take effect immediately explicitly attach to the instance using CLP and then issue the CLP command:

```
UPDATE DATABASE MANAGER CONFIGURATION CLP USING DFT_MON_LOCK ON
```

The default setting can also be turned on through the ADMIN_CMD stored procedure. For example:

```
CALL SYSPROC.ADMIN_CMD('update dbm cfg using DFT_MON_LOCK ON')
```

If the ADMIN_CMD stored procedure is used or if the clp command is used without having previously attached to the instance, the instance must be recycled before the change takes effect.

Information returned

Table 298. Information returned by the SNAPLOCKWAIT administrative view and the SNAP_GET_LOCKWAIT table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| SUBSECTION_NUMBER | BIGINT | ss_number - Subsection number |

Table 298. Information returned by the SNAPLOCKWAIT administrative view and the SNAP_GET_LOCKWAIT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|-------------|--|
| LOCK_MODE | VARCHAR(10) | lock_mode - Lock mode. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |
| LOCK_OBJECT_TYPE | VARCHAR(18) | lock_object_type - Lock object type waited on. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • AUTORESIZE_LOCK • AUTOSTORAGE_LOCK • BLOCK_LOCK • EOT_LOCK • INPLACE_REORG_LOCK • INTERNAL_LOCK • INTERNALB_LOCK • INTERNALC_LOCK • INTERNALJ_LOCK • INTERNALL_LOCK • INTERNALO_LOCK • INTERNALQ_LOCK • INTERNALP_LOCK • INTERNALS_LOCK • INTERNALT_LOCK • INTERNALV_LOCK • KEYVALUE_LOCK • ROW_LOCK • SYSBOOT_LOCK • TABLE_LOCK • TABLE_PART_LOCK • TABLESPACE_LOCK • XML_PATH_LOCK |
| AGENT_ID_HOLDING_LK | BIGINT | agent_id_holding_lock - Agent ID holding lock |

Table 298. Information returned by the SNAPLOCKWAIT administrative view and the SNAP_GET_LOCKWAIT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|--------------|---|
| LOCK_WAIT_START_TIME | TIMESTAMP | lock_wait_start_time - Lock wait start timestamp |
| LOCK_MODE_REQUESTED | VARCHAR(10) | lock_mode_requested - Lock mode requested. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock escalation |
| TABNAME | VARCHAR(128) | table_name - Table name |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| APPL_ID_HOLDING_LK | VARCHAR(128) | appl_id_holding_lk - Application ID holding lock |
| LOCK_ATTRIBUTES | VARCHAR(128) | lock_attributes - Lock attributes. This interface returns a text identifier based on the defines in sqlmon.h. If there are no locks, the text identifier is NONE, otherwise, it is any combination of the following separated by a '+' sign: <ul style="list-style-type: none"> • ALLOW_NEW • DELETE_IN_BLOCK • ESCALATED • INSERT • NEW_REQUEST • RR • RR_IN_BLOCK • UPDATE_DELETE • WAIT_FOR_AVAIL |

Table 298. Information returned by the SNAPLOCKWAIT administrative view and the SNAP_GET_LOCKWAIT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|--|
| LOCK_CURRENT_MODE | VARCHAR(10) | lock_current_mode - Original lock mode before conversion. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |
| LOCK_NAME | VARCHAR(32) | lock_name - Lock name |
| LOCK_RELEASE_FLAGS | BIGINT | lock_release_flags - Lock release flags. |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition identifier. For a non-partitioned table, this element is NULL. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPTBSP_RANGE administrative view and SNAP_GET_TBSP_RANGE table function - Retrieve range snapshot information

The SNAPTBSP_RANGE administrative view and the SNAP_GET_TBSP_RANGE table function return information from a range snapshot.

Important: The SNAPTBSP_RANGE administrative view and SNAP_GET_TBSP_RANGE table function are deprecated and have been replaced by “MON_GET_TABLESPACE_RANGE table function - Get information about table space ranges” on page 820.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPTBSP_RANGE administrative view” on page 1378
- “SNAP_GET_TBSP_RANGE table function” on page 1379

SNAPTbsp_RANGE administrative view

This administrative view allows you to retrieve range snapshot information for the currently connected database.

Used with the SNAPTbsp_QUIESCER administrative view and the MON_GET_TABLESPACE and the MON_GET_CONTAINER table functions, the SNAPTbsp_RANGE administrative view provides information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 299 on page 1381 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPTbsp_RANGE administrative view
- CONTROL privilege on the SNAPTbsp_RANGE administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_TBSP_RANGE table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Select information about table space ranges for all database members for the currently connected database.

```
SELECT TBSP_ID, SUBSTR(TBSP_NAME, 1, 15) AS TBSP_NAME, RANGE_NUMBER,  
       RANGE_STRIPE_SET_NUMBER, RANGE_OFFSET, RANGE_MAX_PAGE,  
       RANGE_MAX_EXTENT, RANGE_START_STRIPE, RANGE_END_STRIPE,  
       RANGE_ADJUSTMENT, RANGE_NUM_CONTAINER, RANGE_CONTAINER_ID,  
       DBPARTITIONNUM FROM SYSIBMADM.SNAPTbsp_RANGE  
ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| TBSP_ID | TBSP_NAME | RANGE_NUMBER | RANGE_STRIPE_SET_NUMBER | ... |
|---------|--------------|--------------|-------------------------|-----|
| 0 | SYSCATSPACE | 0 | 0 | ... |
| 2 | USERSPACE1 | 0 | 0 | ... |
| 3 | SYSTOOLSPACE | 0 | 0 | ... |
| 2 | USERSPACE1 | 0 | 0 | ... |
| 2 | USERSPACE1 | 0 | 0 | ... |

5 record(s) selected.

Output from this query (continued).

| ... | RANGE_OFFSET | RANGE_MAX_PAGE | RANGE_MAX_EXTENT | ... |
|-----|--------------|----------------|------------------|-----|
| ... | 0 | 11515 | 2878 | ... |
| ... | 0 | 479 | 14 | ... |
| ... | 0 | 251 | 62 | ... |
| ... | 0 | 479 | 14 | ... |
| ... | 0 | 479 | 14 | ... |

Output from this query (continued).

| ... | RANGE_START_STRIPE | RANGE_END_STRIPE | RANGE_ADJUSTMENT | ... |
|-----|--------------------|------------------|------------------|-----|
| ... | 0 | 2878 | 0 | ... |
| ... | 0 | 14 | 0 | ... |
| ... | 0 | 62 | 0 | ... |
| ... | 0 | 14 | 0 | ... |
| ... | 0 | 14 | 0 | ... |

Output from this query (continued).

| ... | RANGE_NUM_CONTAINER | RANGE_CONTAINER_ID | DBPARTITIONNUM |
|-----|---------------------|--------------------|----------------|
| ... | 1 | 0 | 0 |
| ... | 1 | 0 | 0 |
| ... | 1 | 0 | 0 |
| ... | 1 | 0 | 1 |
| ... | 1 | 0 | 2 |

SNAP_GET_TBSP_RANGE table function

The SNAP_GET_TBSP_RANGE table function returns the same information as the SNAPTBSP_RANGE administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_TBSP, SNAP_GET_TBSP_PART, SNAP_GET_TBSP QUIESCER and SNAP_GET_CONTAINER table functions, the SNAP_GET_TBSP_RANGE table function provides information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 299 on page 1381 for a complete list of information that can be returned.

Syntax

```

>> SNAP_GET_TBSP_RANGE ( ( dbname [ , member ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TBSP_RANGE table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_TBSP_RANGE table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Select information about the table space range for the table space with `tbasp_id = 2` on the currently connected database member.

```
SELECT TBSP_ID, SUBSTR(TBSP_NAME, 1, 15) AS TBSP_NAME, RANGE_NUMBER,  
       RANGE_STRIPE_SET_NUMBER, RANGE_OFFSET, RANGE_MAX_PAGE, RANGE_MAX_EXTENT,  
       RANGE_START_STRIPE, RANGE_END_STRIPE, RANGE_ADJUSTMENT,  
       RANGE_NUM_CONTAINER, RANGE_CONTAINER_ID  
FROM TABLE(SNAP_GET_TBSP_RANGE('',-1)) AS T WHERE TBSP_ID = 2
```

The following is an example of output from this query.

```

TBSP_ID      TBSP_NAME      RANGE_NUMBER      ...
-----
...          2 USERSPACE1          0 ...

1 record(s) selected.

```

Output from this query (continued).

```

... RANGE_STRIPE_SET_NUMBER RANGE_OFFSET      RANGE_MAX_PAGE      ...
... -----
...          0          0          3967 ...

```

Output from this query (continued).

```

... RANGE_MAX_EXTENT      RANGE_START_STRIPE      RANGE_END_STRIPE      ...
... -----
...          123          0          123 ...

```

Output from this query (continued).

```

... RANGE_ADJUSTMENT      RANGE_NUM_CONTAINER      RANGE_CONTAINER_ID
... -----
...          0          1          0

```

Information returned

Table 299. Information returned by the `SNAPTbsp_RANGE` administrative view and the `SNAP_GET_TBSP_RANGE` table function

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| RANGE_NUMBER | BIGINT | range_number - Range number |
| RANGE_STRIPE_SET_NUMBER | BIGINT | range_stripe_set_number - Stripe set number |
| RANGE_OFFSET | BIGINT | range_offset - Range offset |
| RANGE_MAX_PAGE | BIGINT | range_max_page_number - Maximum page in range |
| RANGE_MAX_EXTENT | BIGINT | range_max_extent - Maximum extent in range |
| RANGE_START_STRIPE | BIGINT | range_start_stripe - Start stripe |
| RANGE_END_STRIPE | BIGINT | range_end_stripe - End stripe |
| RANGE_ADJUSTMENT | BIGINT | range_adjustment - Range adjustment |
| RANGE_NUM_CONTAINER | BIGINT | range_num_container - Number of containers in range |
| RANGE_CONTAINER_ID | BIGINT | range_container_id - Range container |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

SNAPSTMT administrative view and SNAP_GET_STMT table function - Retrieve statement snapshot information

The SNAPSTMT administrative view and the SNAP_GET_STMT table function return information about SQL or XQuery statements from an application snapshot.

Important: The SNAPSTMT administrative view and the SNAP_GET_STMT table function are deprecated. Use the “MON_GET_ACTIVITY table function - Return a list of activities” on page 514 to retrieve the information returned by these deprecated interfaces.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPSTMT administrative view”
- “SNAP_GET_STMT table function” on page 1383

SNAPSTMT administrative view

This administrative view allows you to retrieve statement snapshot information for the currently connected database.

Used with the MON_GET_AGENT, MON_GET_CONNECTION, MON_GET_MEMORY_POOL, and MON_GET_MEMORY_SET table functions and the MON_CURRENT_SQL and SNAPSUBSECTION administrative views, the SNAPSTMT administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS on database-alias** CLP command, but retrieves data from all database members.

The schema is SYSIBMADM.

Refer to Table 300 on page 1385 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPSTMT administrative view
- CONTROL privilege on the SNAPSTMT administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_STMT table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT

- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve rows read, written and operation performed for statements executed on the currently connected single-member database.

```
SELECT SUBSTR(STMT_TEXT,1,30) AS STMT_TEXT, ROWS_READ, ROWS_WRITTEN,
       STMT_OPERATION FROM SYSIBMADM.SNAPSTMT
```

The following is an example of output from this query.

| STMT_TEXT | ROWS_READ | ROWS_WRITTEN | STMT_OPERATION |
|-----------|-----------|--------------|-----------------|
| ----- | ----- | ----- | ----- |
| - | | 0 | 0 FETCH |
| - | | 0 | 0 STATIC_COMMIT |

2 record(s) selected.

SNAP_GET_STMT table function

The SNAP_GET_STMT table function returns the same information as the SNAPSTMT administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_AGENT, SNAP_GET_AGENT_MEMORY_POOL, SNAP_GET_APPL, SNAP_GET_APPL_INFO and SNAP_GET_SUBSECTION table functions, the SNAP_GET_STMT table function provides information equivalent to the **GET SNAPSHOT FOR ALL APPLICATIONS** CLP command, but retrieves data from all database partitions.

Refer to Table 300 on page 1385 for a complete list of information that can be returned.

Syntax

```

>> SNAP_GET_STMT ( ( dbname [ , member ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_STMT table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_STMT table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve rows read, written and operation performed for statements executed on current database member of currently connected database.

```
SELECT SUBSTR(STMT_TEXT,1,30) AS STMT_TEXT, ROWS_READ,  
       ROWS_WRITTEN, STMT_OPERATION FROM TABLE(SNAP_GET_STMT('','-1)) AS T
```

The following is an example of output from this query.

| STMT_TEXT | ROWS_READ | ... |
|-------------------------------|-----------|-----|
| ----- | ----- | ... |
| update t set a=3 | 0 | ... |
| SELECT SUBSTR(STMT_TEXT,1,30) | 0 | ... |
| - | 0 | ... |
| - | 0 | ... |
| update t set a=2 | 9 | ... |
| | | ... |
| 5 record(s) selected. | | ... |

Output from this query (continued).

| ... ROWS_WRITTEN | STMT_OPERATION |
|------------------|---------------------|
| ... | ----- |
| ... | 0 EXECUTE_IMMEDIATE |

```

...      0  FETCH
...      0  NONE
...      0  NONE
...      1  EXECUTE_IMMEDIATE
...

```

Information returned

Table 300. Information returned by the SNAPSTMT administrative view and the SNAP_GET_STMT table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| NUM_AGENTS | BIGINT | num_agents - Number of agents working on a statement |
| AGENTS_TOP | BIGINT | agents_top - Number of agents created |
| STMT_TYPE | VARCHAR(20) | stmt_type - Statement type. This interface returns a text identifier based on defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • DYNAMIC • NON_STMT • STATIC • STMT_TYPE_UNKNOWN |

Table 300. Information returned by the SNAPSTMT administrative view and the SNAP_GET_STMT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|-------------|--|
| STMT_OPERATION | VARCHAR(20) | stmt_operation/operation - Statement operation. This interface returns a text identifier based on defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • CALL • CLOSE • COMPILE • DESCRIBE • EXECUTE • EXECUTE_IMMEDIATE • FETCH • FREE_LOCATOR • GETAA • GETNEXTCHUNK • GETTA • NONE • OPEN • PREP_COMMIT • PREP_EXEC • PREP_OPEN • PREPARE • REBIND • REDIST • REORG • RUNSTATS • SELECT • SET • STATIC_COMMIT • STATIC_ROLLBACK |
| SECTION_NUMBER | BIGINT | section_number - Section number |
| QUERY_COST_ESTIMATE | BIGINT | query_cost_estimate - Query cost estimate |
| QUERY_CARD_ESTIMATE | BIGINT | query_card_estimate - Query number of rows estimate |
| DEGREE_PARALLELISM | BIGINT | degree_parallelism - Degree of parallelism |
| STMT_SORTS | BIGINT | stmt_sorts - Statement sorts |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total sort time |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal rows deleted |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal rows updated |

Table 300. Information returned by the SNAPSTMT administrative view and the SNAP_GET_STMT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|--------------|---|
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal rows inserted |
| FETCH_COUNT | BIGINT | fetch_count - Number of successful fetches |
| STMT_START | TIMESTAMP | stmt_start - Statement operation start timestamp |
| STMT_STOP | TIMESTAMP | stmt_stop - Statement operation stop timestamp |
| STMT_USR_CPU_TIME_S | BIGINT | stmt_usr_cpu_time - User CPU time used by statement (in seconds)* |
| STMT_USR_CPU_TIME_MS | BIGINT | stmt_usr_cpu_time - User CPU time used by statement (fractional, in microseconds)* |
| STMT_SYS_CPU_TIME_S | BIGINT | stmt_sys_cpu_time - System CPU time used by statement (in seconds)* |
| STMT_SYS_CPU_TIME_MS | BIGINT | stmt_sys_cpu_time - System CPU time used by statement (fractional, in microseconds)* |
| STMT_ELAPSED_TIME_S | BIGINT | stmt_elapsed_time - Most recent statement elapsed time (in seconds)* |
| STMT_ELAPSED_TIME_MS | BIGINT | stmt_elapsed_time - Most recent statement elapsed time (fractional, in microseconds)* |
| BLOCKING_CURSOR | SMALLINT | blocking_cursor - Blocking cursor |
| STMT_NODE_NUMBER | SMALLINT | stmt_node_number - Statement node |
| CURSOR_NAME | VARCHAR(128) | cursor_name - Cursor name |
| CREATOR | VARCHAR(128) | creator - Application creator |
| PACKAGE_NAME | VARCHAR(128) | package_name - Package name |
| STMT_TEXT | CLOB(16 M) | stmt_text - SQL statement text |
| CONSISTENCY_TOKEN | VARCHAR(128) | consistency_token - Package consistency token |
| PACKAGE_VERSION_ID | VARCHAR(128) | package_version_id - Package version |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |

Table 300. Information returned by the SNAPSTMT administrative view and the SNAP_GET_STMT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--|-----------|---|
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads monitor element |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| <p>* To calculate the total time spent for the monitor element that this column is based on, you must add the full seconds reported in the column for this monitor element that ends with _S to the fractional seconds reported in the column for this monitor element that ends with _MS, using the following formula: $(\text{monitor-element-name_S} \times 1,000,000 + \text{monitor-element-name_MS}) \div 1,000,000$. For example, $(\text{ELAPSED_EXEC_TIME_S} \times 1,000,000 + \text{ELAPSED_EXEC_TIME_MS}) \div 1,000,000$.</p> | | |

SNAP_GET_STO_PATHS

The SNAP_GET_STO_PATHS table function returns snapshot information from the storage_paths logical data group.

Note: This table function has been deprecated and replaced by the “ADMIN_GET_STORAGE_PATHS table function - retrieve automatic storage path information” on page 217.

►►—SNAP_GET_STO_PATHS—(—dbname—,—member—)—◄◄

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(255) that specifies a valid database

name in the same instance as the currently connected database when calling this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify the NULL value to take the snapshot from the currently connected database.

member

An input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for all active database partitions. An active database member is a member where the database is available for connection and use by applications.

If the null value is specified, -1 is set implicitly.

If both parameters are set to NULL, the snapshot will be taken only if a file has not previously been created by the SNAP_WRITE_FILE stored procedure for the corresponding snapshot API request type.

Authorization

One of the following authorities is required to execute the function:

- EXECUTE privilege on the function
- DATAACCESS authority

To access snapshot monitor data, one of the following authorities is required:

- SYSMON authority
- SYSCTRL authority
- SYSMANT authority
- SYSADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

The function returns the following table.

Table 301. Information returned by the SNAP_GET_STO_PATHS table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | snapshot_timestamp - Snapshot timestamp |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| DB_STORAGE_PATH | VARCHAR(256) | db_storage_path - Automatic storage path |

SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS_V97 table function - Retrieve automatic storage path information

The SNAPSTORAGE_PATHS administrative view and the SNAP_GET_STORAGE_PATHS_V97 return a list of automatic storage paths for the database including file system information for each storage path, specifically, from the db_storage_group logical data group.

Note: The “SNAPSTORAGE_PATHS administrative view” and the “SNAP_GET_STORAGE_PATHS_V97” on page 1391 have been deprecated and replaced by the “ADMIN_GET_STORAGE_PATHS table function - retrieve automatic storage path information” on page 217. This function and view might be removed in a future release.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPSTORAGE_PATHS administrative view”
- “SNAP_GET_STORAGE_PATHS_V97” on page 1391

SNAPSTORAGE_PATHS administrative view

This administrative view allows you to retrieve automatic storage path information for the currently connected database.

Used with the MON_GET_DATABASE, MON_GET_HADR, MON_GET_MEMORY_POOL, MON_GET_MEMORY_SET, and MON_GET_TRANSACTION_LOG table functions, the SNAPSTORAGE_PATHS administrative view provides information equivalent to the **GET SNAPSHOT FOR DATABASE ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 302 on page 1392 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPSTORAGE_PATHS administrative view
- CONTROL privilege on the SNAPSTORAGE_PATHS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following authorizations is required to use the table function:

- EXECUTE privilege on the SNAP_GET_STORAGE_PATHS_V97 table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

To access snapshot monitor data, one of the following authorities is also required:

- SYSMON authority

- SYSCTRL authority
- SYSMANT authority
- SYSADM authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve the storage path for the currently connected single-member database.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, SUBSTR(DB_STORAGE_PATH,1,8)
      AS DB_STORAGE_PATH, SUBSTR(HOSTNAME,1,10) AS HOSTNAME
FROM SYSIBMADM.SNAPSTORAGE_PATHS
```

The following is an example of output from this query.

| DB_NAME | DB_STORAGE_PATH | HOSTNAME |
|---------|-----------------|----------|
| ----- | ----- | ----- |
| STOPATH | d: | JESSICAE |

1 record(s) selected.

SNAP_GET_STORAGE_PATHS_V97

The SNAP_GET_STORAGE_PATHS_V97 table function returns similar information as the SNAPSTORAGE_PATHS administrative view. It allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DB, SNAP_GET_DETAILLOG, SNAP_GET_HADR and SNAP_GET_DB_MEMORY_POOL table functions, the SNAP_GET_STORAGE_PATHS_V97 table function provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES** CLP command.

Refer to Table 302 on page 1392 for a complete list of information that can be returned.

Syntax

```
▶▶ SNAP_GET_STORAGE_PATHS_V97 ( (—dbname—, member—) ) ▶▶
```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid member number. Specify -1 for the current member, or -2 for an aggregate of all active members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all members where the database is active.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_STORAGE_PATHS_V97 table function takes a snapshot for the currently connected database and database member.

Authorization

- EXECUTE privilege on the SNAP_GET_STORAGE_PATHS_V97 table function.
- DATAACCESS authority

To access snapshot monitor data, one of the following authorities is required:

- SYSMON authority
- SYSCTRL authority
- SYSMANT suthority
- SYSADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Retrieve the storage path information for all active databases.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, DB_STORAGE_PATH
FROM TABLE(SNAP_GET_STORAGE_PATHS_V97(CAST (NULL AS VARCHAR(128)), -1)) AS T
```

The following is an example of output from this query.

```
DB_NAME  DB_STORAGE_PATH
-----  -
STOPATH  /home/jessica/sdb
MYDB     /home/jessica/mdb
```

2 record(s) selected

Information returned

The BUFFERPOOL monitor switch must be turned on in order for the file system information to be returned.

Table 302. Information returned by the SNAPSTORAGE_PATHS administrative view and the SNAP_GET_STORAGE_PATHS_V97 table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |

Table 302. Information returned by the `SNAPSTORAGE_PATHS` administrative view and the `SNAP_GET_STORAGE_PATHS_V97` table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|--------------|---|
| DB_NAME | VARCHAR(128) | db_name - Database name |
| DB_STORAGE_PATH | VARCHAR(256) | db_storage_path - Automatic storage path |
| DB_STORAGE_PATH_WITH_DPE | VARCHAR(256) | db_storage_path_with_dpe - Storage path including database partition expression monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| DB_STORAGE_PATH_STATE | VARCHAR(16) | db_storage_path_state - Storage path state monitor element |
| FS_ID | VARCHAR(22) | fs_id - Unique file system identification number |
| FS_TOTAL_SIZE | BIGINT | fs_total_size - Total size of a file system |
| FS_USED_SIZE | BIGINT | fs_used_size - Amount of space used on a file system |
| STO_PATH_FREE_SIZE | BIGINT | sto_path_free_size - Automatic storage path free space |

SNAPTAB administrative view and SNAP_GET_TAB table function - Retrieve table logical data group snapshot information

The SNAPTAB administrative view and the SNAP_GET_TAB table function return snapshot information from the table logical data group.

Important: The SNAPTAB administrative view and SNAP_GET_TAB table function are deprecated. Use the “MON_GET_TABLE table function - get table metrics” on page 800, to retrieve the information returned by these deprecated interfaces.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPTAB administrative view”
- “SNAP_GET_TAB table function” on page 1394

SNAPTAB administrative view

This administrative view allows you to retrieve table logical data group snapshot information for the currently connected database.

Used in conjunction with the SNAPTAB_REORG administrative view, the SNAPTAB administrative view returns equivalent information to the **GET SNAPSHOT FOR TABLES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 303 on page 1396 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPTAB administrative view
- CONTROL privilege on the SNAPTAB administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_TAB table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve the schema and name for all active tables.

```
SELECT SUBSTR(TABSCHEMA,1,8), SUBSTR(TABNAME,1,15) AS TABNAME, TAB_TYPE,  
       DBPARTITIONNUM FROM SYSIBMADM.SNAPTAB
```

The following is an example of output from this query.

| TABSCHEMA | TABNAME | TAB_TYPE | DBPARTITIONNUM |
|-----------|---------------|------------|----------------|
| SYSTOOLS | HMON_ATM_INFO | USER_TABLE | 0 |

1 record selected.

SNAP_GET_TAB table function

The SNAP_GET_TAB table function returns the same information as the SNAPTAB administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used in conjunction with the SNAP_GET_TAB_REORG table function, the SNAP_GET_TAB table function returns equivalent information to the **GET SNAPSHOT FOR TABLES ON database-alias** CLP command.

Syntax

The schema is SYSPROC.

dbname

member

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TAB table function takes a snapshot for the currently connected database and database member number.

One of the following authorizations is required:

- In addition, to access snapshot monitor data, one of the following authorities is also required:

- ## Default PUBLIC privilege

Built-in routines and views 1395

Example

Retrieve a list of active tables as an aggregate view for the currently connected database.

```
SELECT SUBSTR(TABSCHEMA,1,8) AS TABSCHEMA, SUBSTR(TABNAME,1,15) AS TABNAME,  
       TAB_TYPE, DBPARTITIONNUM FROM TABLE(SNAP_GET_TAB('','-2')) AS T
```

The following is an example of output from this query.

| TABSCHEMA | TABNAME | TAB_TYPE | DBPARTITIONNUM |
|-----------|---------------|------------|----------------|
| SYSTOOLS | HMON_ATM_INFO | USER_TABLE | - |
| JESSICAE | EMPLOYEE | USER_TABLE | - |

Information returned

Table 303. Information returned by the SNAPTAB administrative view and the SNAP_GET_TAB table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TABNAME | VARCHAR(128) | table_name - Table name |
| TAB_FILE_ID | BIGINT | table_file_id - Table file identification |
| TAB_TYPE | VARCHAR(14) | table_type - Table type. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none">• CATALOG_TABLE• DROPPED_TABLE• REORG_TABLE• SYNOPSIS_TABLE• TEMP_TABLE• USER_TABLE |
| DATA_OBJECT_PAGES | BIGINT | data_object_pages - Data object pages |
| INDEX_OBJECT_PAGES | BIGINT | index_object_pages - Index object pages |
| LOB_OBJECT_PAGES | BIGINT | lob_object_pages - LOB object pages |
| LONG_OBJECT_PAGES | BIGINT | long_object_pages - Long object pages |
| XDA_OBJECT_PAGES | BIGINT | xda_object_pages - XDA Object Pages |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| OVERFLOW_ACCESSES | BIGINT | overflow_accesses - Accesses to overflowed records |
| PAGE_REORGS | BIGINT | page_reorgs - Page reorganizations |

Table 303. Information returned by the SNAPTAB administrative view and the SNAP_GET_TAB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------|-----------|--|
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition identifier. For a non-partitioned table, this element will be NULL. |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAP_GET_TAB_V91

The SNAP_GET_TAB_V91 table function returns snapshot information from the table logical data group.

Important: The SNAP_GET_TAB_V91 table function is deprecated. Use the “MON_GET_TABLE table function - get table metrics” on page 800 to retrieve the information returned by this deprecated interface.

Refer to Table 304 on page 1398 for a complete list of information that can be returned.

Syntax

```

--> SNAP_GET_TAB_V91 ( (dbname
                        |, dbpartitionnum
                        ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

dbpartitionnum

An optional input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for an aggregate of all active database partitions. If *dbname* is not set to NULL and *dbpartitionnum* is set to NULL, -1 is set implicitly for *dbpartitionnum*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database partitions. An active database partition is a partition where the database is available for connection and use by applications.

If both *dbname* and *dbpartitionnum* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If

a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TAB_V91 table function takes a snapshot for the currently connected database and database partition number.

Authorization

- One of the following authorizations is required:
- EXECUTE privilege on the SNAP_GET_TAB_V91 table function
 - DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of active tables as an aggregate view for the currently connected database.

```
SELECT SUBSTR(TABSCHEMA,1,8) AS TABSCHEMA, SUBSTR(TABNAME,1,15) AS TABNAME,
       TAB_TYPE, DBPARTITIONNUM FROM TABLE(SNAP_GET_TAB('',-2)) AS T
```

The following is an example of output from this query.

| TABSCHEMA | TABNAME | TAB_TYPE | DBPARTITIONNUM |
|-----------|---------------|------------|----------------|
| SYSTOOLS | HMON_ATM_INFO | USER_TABLE | - |
| JESSICAE | EMPLOYEE | USER_TABLE | - |

Information returned

Table 304. Information returned by the SNAP_GET_TAB_V91 table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TABNAME | VARCHAR(128) | table_name - Table name |
| TAB_FILE_ID | BIGINT | table_file_id - Table file identification |

Table 304. Information returned by the SNAP_GET_TAB_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|--|
| TAB_TYPE | VARCHAR(14) | table_type - Table type. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • USER_TABLE • DROPPED_TABLE • TEMP_TABLE • CATALOG_TABLE • REORG_TABLE |
| DATA_OBJECT_PAGES | BIGINT | data_object_pages - Data object pages |
| INDEX_OBJECT_PAGES | BIGINT | index_object_pages - Index object pages |
| LOB_OBJECT_PAGES | BIGINT | lob_object_pages - LOB object pages |
| LONG_OBJECT_PAGES | BIGINT | long_object_pages - Long object pages |
| XDA_OBJECT_PAGES | BIGINT | xda_object_pages - XDA Object Pages |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| OVERFLOW_ACCESSES | BIGINT | overflow_accesses - Accesses to overflowed records |
| PAGE_REORGS | BIGINT | page_reorgs - Page reorganizations |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition identifier. For a non-partitioned table, this element will be NULL. |

SNAPTBSP administrative view and SNAP_GET_TBSP table function - Retrieve table space logical data group snapshot information

The SNAPTBSP administrative view and the SNAP_GET_TBSP table function return snapshot information from the table space logical data group.

Important: The SNAPTBSP administrative view and the SNAP_GET_TBSP table function are deprecated and have been replaced by the “MON_GET_TABLESPACE table function - Get table space metrics” on page 805.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPTBSP administrative view” on page 1400
- “SNAP_GET_TBSP table function” on page 1401

SNAPTBSP administrative view

This administrative view allows you to retrieve table space logical data group snapshot information for the currently connected database.

Used in conjunction with the SNAPTBSP_PART, SNAPTBSP QUIESCER, SNAPTBSP_RANGE, SNAPCONTAINER administrative views, the SNAPTBSP administrative view returns information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 305 on page 1402 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPTBSP administrative view
- CONTROL privilege on the SNAPTBSP administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_TBSP table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of table spaces on the catalog database member for the currently connected database.

```
SELECT SUBSTR(TBSP_NAME,1,30) AS TBSP_NAME, TBSP_ID, TBSP_TYPE,  
       TBSP_CONTENT_TYPE FROM SYSIBMADM.SNAPTBSP WHERE DBPARTITIONNUM = 1
```

The following is an example of output from this query.

| TBSP_NAME | TBSP_ID | TBSP_TYPE | TBSP_CONTENT_TYPE |
|------------|---------|-----------|-------------------|
| TEMPSPACE1 | 1 | SMS | SYSTEMP |
| USERSPACE1 | 2 | DMS | LONG |

2 record(s) selected.

SNAP_GET_TBSP table function

The SNAP_GET_TBSP table function returns the same information as the SNAPTBSP administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used in conjunction with the SNAP_GET_TBSP_PART, SNAP_GET_TBSP_QUIESCER, SNAP_GET_TBSP_RANGE, SNAP_GET_CONTAINER table functions, the SNAP_GET_TBSP table function returns information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 305 on page 1402 for a complete list of information that can be returned.

Syntax

```

▶▶ SNAP_GET_TBSP ( ( dbname [ , member ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An input argument of type INTEGER that specifies a valid member number. Specify -1 for the current member. If the null value is specified, -1 is set implicitly.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TBSP table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_TBSP table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of table spaces for all database members for the currently connected database.

```
SELECT SUBSTR(TBSP_NAME,1,10) AS TBSP_NAME, TBSP_ID, TBSP_TYPE,
       TBSP_CONTENT_TYPE, DBPARTITIONNUM FROM TABLE(SNAP_GET_TBSP('')) AS T
```

The following is an example of output from this query.

| TBSP_NAME | TBSP_ID | TBSP_TYPE | TBSP_CONTENT_TYPE | DBPARTITIONNUM |
|------------|---------|-----------|-------------------|----------------|
| TEMPSPACE1 | 1 | SMS | SYSTEMP | 1 |
| USERSPACE1 | 2 | DMS | LONG | 1 |
| SYSCATSPAC | 0 | DMS | ANY | 0 |
| TEMPSPACE1 | 1 | SMS | SYSTEMP | 0 |
| USERSPACE1 | 2 | DMS | LONG | 0 |
| SYSTOOLSPA | 3 | DMS | LONG | 0 |
| TEMPSPACE1 | 1 | SMS | SYSTEMP | 2 |
| USERSPACE1 | 2 | DMS | LONG | 2 |

8 record(s) selected.

Information returned

Table 305. Information returned by the SNAPTBSP administrative view and the SNAP_GET_TBSP table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| TBSP_TYPE | VARCHAR(10) | tablespace_type - Table space type. This interface returns a text identifier based on defines in sqlutil.h, and is one of: <ul style="list-style-type: none"> • DMS • SMS |

Table 305. Information returned by the SNAPTBSP administrative view and the SNAP_GET_TBSP table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-------------|---|
| TBSP_CONTENT_TYPE | VARCHAR(10) | tablespace_content_type - Table space contents type. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • ANY • LARGE • SYSTEMP • USRTEMP |
| TBSP_PAGE_SIZE | BIGINT | tablespace_page_size - Table space page size |
| TBSP_EXTENT_SIZE | BIGINT | tablespace_extent_size - Table space extent size |
| TBSP_PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table space prefetch size |
| TBSP_CUR_POOL_ID | BIGINT | tablespace_cur_pool_id - Buffer pool currently being used |
| TBSP_NEXT_POOL_ID | BIGINT | tablespace_next_pool_id - Buffer pool that will be used at next startup |
| FS_CACHING | SMALLINT | fs_caching - File system caching |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads |

Table 305. Information returned by the SNAPTBSP administrative view and the SNAP_GET_TBSP table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|---|
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer pool asynchronous read time |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests |
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer pool no victim buffers |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |

Table 305. Information returned by the *SNAPTBSP* administrative view and the *SNAP_GET_TBSP* table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|-------------|--|
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| FILES_CLOSED | BIGINT | files_closed - Database files closed |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| TBSP_REBALANCER_MODE | VARCHAR(10) | tablespace_rebalancer_mode - Rebalancer mode. This interface returns a text identifier based on defines in <code>sqlmon.h</code> , and is one of: <ul style="list-style-type: none"> • NO_REBAL • FWD_REBAL • REV_REBAL |
| TBSP_USING_AUTO_STORAGE | SMALLINT | tablespace_using_auto_storage - Table space enabled for automatic storage |
| TBSP_AUTO_RESIZE_ENABLED | SMALLINT | tablespace_auto_resize_enabled - Table space automatic resizing enabled |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPTBSP_PART administrative view and SNAP_GET_TBSP_PART table function - Retrieve tablespace_nodeinfo logical data group snapshot information

The *SNAPTBSP_PART* administrative view and the *SNAP_GET_TBSP_PART* table function return snapshot information from the *tablespace_nodeinfo* logical data group.

Important: The *SNAPTBSP_PART* administrative view and the *SNAP_GET_TBSP_PART* table function are deprecated and have been replaced by the “*MON_GET_TABLESPACE* table function - Get table space metrics” on page 805.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “*SNAPTBSP_PART* administrative view” on page 1406
- “*SNAP_GET_TBSP_PART* table function” on page 1407

SNAPTBSP_PART administrative view

This administrative view allows you to retrieve tablespace_nodeinfo logical data group snapshot information for the currently connected database.

Used in conjunction with the SNAPTBSP, SNAPTBSP_QUIESCER, SNAPTBSP_RANGE, SNAPCONTAINER administrative views, the SNAPTBSP_PART administrative view returns information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 306 on page 1408 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPTBSP_RANGE administrative view
- CONTROL privilege on the SNAPTBSP_RANGE administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following authorizations is required to use the table function:

- EXECUTE privilege on the SNAP_GET_TBSP_PART table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of table spaces and their state for all database partitions of the currently connected database.

```
SELECT SUBSTR(TBSP_NAME,1,30) AS TBSP_NAME, TBSP_ID,  
       SUBSTR(TBSP_STATE,1,30) AS TBSP_STATE, DBPARTITIONNUM  
FROM SYSIBMADM.SNAPTBSP_PART
```

The following is an example of output from this query.

| TBSP_NAME | TBSP_ID | TBSP_STATE | DBPARTITIONNUM |
|-------------|---------|------------|----------------|
| SYSCATSPACE | 0 | NORMAL | 0 |
| TEMPSPACE1 | 1 | NORMAL | 0 |
| USERSPACE1 | 2 | NORMAL | 0 |
| TEMPSPACE1 | 1 | NORMAL | 1 |
| USERSPACE1 | 2 | NORMAL | 1 |

5 record(s) selected.

SNAP_GET_TBSP_PART table function

The SNAP_GET_TBSP_PART table function returns the same information as the SNAPTbsp_PART administrative view, but allows you to retrieve the information for a specific database on a specific database partition, aggregate of all database partitions or all database partitions.

Used in conjunction with the SNAP_GET_TBSP, SNAP_GET_TBSP QUIESCER, SNAP_GET_TBSP_RANGE, SNAP_GET_CONTAINER table functions, the SNAP_GET_TBSP_PART table function returns information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 306 on page 1408 for a complete list of information that can be returned.

Syntax

```

▶▶ SNAP_GET_TBSP_PART ( ( dbname [ , member ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid member number. Specify -1 for the current member, or -2 for an aggregate of all active members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all members where the database is active.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TBSP_PART table function takes a snapshot for the currently connected database and member.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_TBSP_PART table function.
- DATAACCESS authority

To access snapshot monitor data, one of the following authorities is also required:

- SYSMON authority
- SYSCTRL authority
- SYSMANT authority
- SYSADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of table spaces and their state for the connected database partition of the connected database.

```
SELECT SUBSTR(TBSP_NAME,1,30) AS TBSP_NAME, TBSP_ID,  
       SUBSTR(TBSP_STATE,1,30) AS TBSP_STATE  
FROM TABLE(SNAP_GET_TBSP_PART(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

| TBSP_NAME | TBSP_ID | TBSP_STATE |
|------------------|---------|------------|
| SYSCATSPACE | | 0 NORMAL |
| TEMPSPACE1 | | 1 NORMAL |
| USERSPACE1 | | 2 NORMAL |
| SYSTOOLSPACE | | 3 NORMAL |
| SYSTOOLSTMPSPACE | | 4 NORMAL |

5 record(s) selected.

Information returned

Table 306. Information returned by the SNAPTbsp_PART administrative view and the SNAP_GET_TBSP_PART table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |

Table 306. Information returned by the SNAPTBSP_PART administrative view and the SNAP_GET_TBSP_PART table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------------|------------------|---|
| TBSP_STATE | VARCHAR
(256) | <p>tablespace_state - Table space state. This interface returns a text identifier based on defines in sqlutil.h and is combination of the following separated by a '+' sign:</p> <ul style="list-style-type: none"> • BACKUP_IN_PROGRESS • BACKUP_PENDING • DELETE_PENDING • DISABLE_PENDING • DROP_PENDING • LOAD_IN_PROGRESS • LOAD_PENDING • NORMAL • OFFLINE • PSTAT_CREATION • PSTAT_DELETION • QUIESCED_EXCLUSIVE • QUIESCED_SHARE • QUIESCED_UPDATE • REBAL_IN_PROGRESS • REORG_IN_PROGRESS • RESTORE_IN_PROGRESS • RESTORE_PENDING • ROLLFORWARD_IN_PROGRESS • ROLLFORWARD_PENDING • STORDEF_ALLOWED • STORDEF_CHANGED • STORDEF_FINAL_VERSION • STORDEF_PENDING • SUSPEND_WRITE |
| TBSP_PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table space prefetch size |
| TBSP_NUM_QUIESCERS | BIGINT | tablespace_num_quiescers - Number of quiescers |
| TBSP_STATE_CHANGE_OBJECT_ID | BIGINT | tablespace_state_change_object_id - State change object identification |
| TBSP_STATE_CHANGE_TBSP_ID | BIGINT | tablespace_state_change_ts_id - State change table space identification |
| TBSP_MIN_RECOVERY_TIME | TIMESTAMP | tablespace_min_recovery_time - Minimum recovery time for rollforward |
| TBSP_TOTAL_PAGES | BIGINT | tablespace_total_pages - Total pages in table space |
| TBSP_USABLE_PAGES | BIGINT | tablespace_usable_pages - Usable pages in table space |
| TBSP_USED_PAGES | BIGINT | tablespace_used_pages - Used pages in table space |

Table 306. Information returned by the *SNAPTbsp_PART* administrative view and the *SNAP_GET_TBSP_PART* table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|--------------|--|
| TBSP_FREE_PAGES | BIGINT | tablespace_free_pages - Free pages in table space |
| TBSP_PENDING_FREE_PAGES | BIGINT | tablespace_pending_free_pages - Pending free pages in table space |
| TBSP_PAGE_TOP | BIGINT | tablespace_page_top - Table space high water mark |
| REBALANCER_MODE | VARCHAR (30) | tablespace_rebalancer_mode - Rebalancer mode. This interface returns a text identifier based on defines in <i>sqlmon.h</i> , and is one of: <ul style="list-style-type: none"> • FWD_REBAL • NO_REBAL • REV_REBAL • FWD_REBAL_OF_2PASS • REV_REBAL_OF_2PASS |
| REBALANCER_EXTENTS_REMAINING | BIGINT | tablespace_rebalancer_extents_remaining - Total number of extents to be processed by the rebalancer |
| REBALANCER_EXTENTS_PROCESSED | BIGINT | tablespace_rebalancer_extents_processed - Number of extents the rebalancer has processed |
| REBALANCER_PRIORITY | BIGINT | tablespace_rebalancer_priority - Current rebalancer priority |
| REBALANCER_START_TIME | TIMESTAMP | tablespace_rebalancer_start_time - Rebalancer start time |
| REBALANCER_RESTART_TIME | TIMESTAMP | tablespace_rebalancer_restart_time - Rebalancer restart time |
| REBALANCER_LAST_EXTENT_MOVED | BIGINT | tablespace_rebalancer_last_extent_moved - Last extent moved by the rebalancer |
| TBSP_NUM_RANGES | BIGINT | tablespace_num_ranges - Number of ranges in the table space map |
| TBSP_NUM_CONTAINERS | BIGINT | tablespace_num_containers - Number of containers in table space |
| TBSP_INITIAL_SIZE | BIGINT | tablespace_initial_size - Initial table space size |
| TBSP_CURRENT_SIZE | BIGINT | tablespace_current_size - Current table space size |
| TBSP_MAX_SIZE | BIGINT | tablespace_max_size - Maximum table space size |
| TBSP_INCREASE_SIZE | BIGINT | tablespace_increase_size - Increase size in bytes |
| TBSP_INCREASE_SIZE_PERCENT | SMALLINT | tablespace_increase_size_percent - Increase size by percent |
| TBSP_LAST_RESIZE_TIME | TIMESTAMP | tablespace_last_resize_time - Time of last successful resize |
| TBSP_LAST_RESIZE_FAILED | SMALLINT | tablespace_last_resize_failed - Last resize attempt failed |

Table 306. Information returned by the SNAPTBSP_PART administrative view and the SNAP_GET_TBSP_PART table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| TBSP_PATHS_DROPPED | SMALLINT | Indicates that the table space resides on one or more storage paths that have been dropped (0 - No, 1 - Yes) |

SNAP_GET_TBSP_PART_V97 table function - Retrieve tablespace_nodeinfo logical data group snapshot information

The SNAP_GET_TBSP_PART_V97 table function returns tablespace_nodeinfo logical data group snapshot information for a specific database on a specific database member, aggregate of all database members or all database members.

Important: The SNAP_GET_TBSP_PART_V97 table function is deprecated and has been replaced by the “MON_GET_TABLESPACE table function - Get table space metrics” on page 805.

Refer to Table 307 on page 1412 for a complete list of information that can be returned.

Syntax

```

▶▶ SNAP_GET_TBSP_PART_V97 ( ( dbname [ , member ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TBSP_PART_V97 table function takes a snapshot for the currently

connected database and database member number.

Authorization

- EXECUTE privilege on the SNAP_GET_TBSP_PART_V97 table function.
- DATAACCESS authority

To access snapshot monitor data, one of the following authorities is required:

- SYSMON authority
- SYSCTRL authority
- SYSMANT authority
- SYSADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of table spaces and their state for the connected database member of the connected database.

```
SELECT SUBSTR(TBSP_NAME,1,30) AS TBSP_NAME, TBSP_ID,  
SUBSTR(TBSP_STATE,1,30) AS TBSP_STATE  
FROM TABLE(SNAP_GET_TBSP_PART_V97(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following output is an example of sample output from this query.

| TBSP_NAME | TBSP_ID | TBSP_STATE |
|------------------|---------|------------|
| ----- | ----- | ----- |
| SYSCATSPACE | | 0 NORMAL |
| TEMPSPACE1 | | 1 NORMAL |
| USERSPACE1 | | 2 NORMAL |
| SYSTOOLSPACE | | 3 NORMAL |
| SYSTOOLSTMPSPACE | | 4 NORMAL |

5 record(s) selected.

Information returned

Table 307. Information returned by the SNAP_GET_TBSP_PART_V97 table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|---------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_NAME | VARCHAR (128) | tablespace_name - Table space name |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |

Table 307. Information returned by the SNAP_GET_TBSP_PART_V97 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------------|------------------|---|
| TBSP_STATE | VARCHAR
(256) | <p>tablespace_state - Table space state. This interface returns a text identifier based on defines in sqlutil.h and is combination of the following separated by a '+' sign:</p> <ul style="list-style-type: none"> • BACKUP_IN_PROGRESS • BACKUP_PENDING • DELETE_PENDING • DISABLE_PENDING • DROP_PENDING • LOAD_IN_PROGRESS • LOAD_PENDING • NORMAL • OFFLINE • PSTAT_CREATION • PSTAT_DELETION • QUIESCED_EXCLUSIVE • QUIESCED_SHARE • QUIESCED_UPDATE • REBAL_IN_PROGRESS • REORG_IN_PROGRESS • RESTORE_IN_PROGRESS • RESTORE_PENDING • ROLLFORWARD_IN_PROGRESS • ROLLFORWARD_PENDING • STORDEF_ALLOWED • STORDEF_CHANGED • STORDEF_FINAL_VERSION • STORDEF_PENDING • SUSPEND_WRITE |
| TBSP_PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table space prefetch size |
| TBSP_NUM_QUIESCERS | BIGINT | tablespace_num_quiescers - Number of quiescers |
| TBSP_STATE_CHANGE_OBJECT_ID | BIGINT | tablespace_state_change_object_id - State change object identification |
| TBSP_STATE_CHANGE_TBSP_ID | BIGINT | tablespace_state_change_ts_id - State change table space identification |
| TBSP_MIN_RECOVERY_TIME | TIMESTAMP | tablespace_min_recovery_time - Minimum recovery time for rollforward |
| TBSP_TOTAL_PAGES | BIGINT | tablespace_total_pages - Total pages in table space |
| TBSP_USABLE_PAGES | BIGINT | tablespace_usable_pages - Usable pages in table space |
| TBSP_USED_PAGES | BIGINT | tablespace_used_pages - Used pages in table space |

Table 307. Information returned by the `SNAP_GET_TBSP_PART_V97` table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|--------------|--|
| TBSP_FREE_PAGES | BIGINT | tablespace_free_pages - Free pages in table space |
| TBSP_PENDING_FREE_PAGES | BIGINT | tablespace_pending_free_pages - Pending free pages in table space |
| TBSP_PAGE_TOP | BIGINT | tablespace_page_top - Table space high water mark |
| REBALANCER_MODE | VARCHAR (30) | tablespace_rebalancer_mode - Rebalancer mode. This interface returns a text identifier based on defines in <code>sqlmon.h</code> , and is one of: <ul style="list-style-type: none"> • FWD_REBAL • NO_REBAL • REV_REBAL • FWD_REBAL_OF_2PASS • REV_REBAL_OF_2PASS |
| REBALANCER_EXTENTS_REMAINING | BIGINT | tablespace_rebalancer_extents_remaining - Total number of extents to be processed by the rebalancer |
| REBALANCER_EXTENTS_PROCESSED | BIGINT | tablespace_rebalancer_extents_processed - Number of extents the rebalancer has processed |
| REBALANCER_PRIORITY | BIGINT | tablespace_rebalancer_priority - Current rebalancer priority |
| REBALANCER_START_TIME | TIMESTAMP | tablespace_rebalancer_start_time - Rebalancer start time |
| REBALANCER_RESTART_TIME | TIMESTAMP | tablespace_rebalancer_restart_time - Rebalancer restart time |
| REBALANCER_LAST_EXTENT_MOVED | BIGINT | tablespace_rebalancer_last_extent_moved - Last extent moved by the rebalancer |
| TBSP_NUM_RANGES | BIGINT | tablespace_num_ranges - Number of ranges in the table space map |
| TBSP_NUM_CONTAINERS | BIGINT | tablespace_num_containers - Number of containers in table space |
| TBSP_INITIAL_SIZE | BIGINT | tablespace_initial_size - Initial table space size |
| TBSP_CURRENT_SIZE | BIGINT | tablespace_current_size - Current table space size |
| TBSP_MAX_SIZE | BIGINT | tablespace_max_size - Maximum table space size |
| TBSP_INCREASE_SIZE | BIGINT | tablespace_increase_size - Increase size in bytes |
| TBSP_INCREASE_SIZE_PERCENT | SMALLINT | tablespace_increase_size_percent - Increase size by percent |
| TBSP_LAST_RESIZE_TIME | TIMESTAMP | tablespace_last_resize_time - Time of last successful resize |
| TBSP_LAST_RESIZE_FAILED | SMALLINT | tablespace_last_resize_failed - Last resize attempt failed |

Table 307. Information returned by the SNAP_GET_TBSP_PART_V97 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| TBSP_PATHS_DROPPED | SMALLINT | Indicates that the table space resides on one or more storage paths that have been dropped (0 - No, 1 - Yes) |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

SNAP_GET_TBSP_V91

The SNAP_GET_TBSP_V91 table function returns snapshot information from the table space logical data group.

Important: The SNAP_GET_TBSP_V91 table function is deprecated and have been replaced by the “MON_GET_TABLESPACE table function - Get table space metrics” on page 805.

Refer to Table 308 on page 1416 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_TBSP_V91 (—dbname —, dbpartitionnum —) —►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

dbpartitionnum

An optional input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for an aggregate of all active database partitions. If *dbname* is not set to NULL and *dbpartitionnum* is set to NULL, -1 is set implicitly for *dbpartitionnum*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database partitions. An active database partition is a partition where the database is available for connection and use by applications.

If both *dbname* and *dbpartitionnum* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TBSP_V91 table function takes a snapshot for the currently connected database and database partition number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_TBSP_V91 table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMOINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of table spaces for all database partitions for the currently connected database.

```
SELECT SUBSTR(TBSP_NAME,1,10) AS TBSP_NAME, TBSP_ID, TBSP_TYPE,  
       TBSP_CONTENT_TYPE, DBPARTITIONNUM FROM TABLE(SNAP_GET_TBSP_V91('')) AS T
```

The following is an example of output from this query.

| TBSP_NAME | TBSP_ID | TBSP_TYPE | TBSP_CONTENT_TYPE | DBPARTITIONNUM |
|------------|---------|-----------|-------------------|----------------|
| TEMPSPACE1 | 1 | SMS | SYSTEMP | 1 |
| USERSPACE1 | 2 | DMS | LONG | 1 |
| SYSCATSPAC | 0 | DMS | ANY | 0 |
| TEMPSPACE1 | 1 | SMS | SYSTEMP | 0 |
| USERSPACE1 | 2 | DMS | LONG | 0 |
| SYSTOOLSPA | 3 | DMS | LONG | 0 |
| TEMPSPACE1 | 1 | SMS | SYSTEMP | 2 |
| USERSPACE1 | 2 | DMS | LONG | 2 |

8 record(s) selected.

Information returned

Table 308. Information returned by the SNAP_GET_TBSP_V91 table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |

Table 308. Information returned by the SNAP_GET_TBSP_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-------------|---|
| TBSP_TYPE | VARCHAR(10) | tablespace_type - Table space type. This interface returns a text identifier based on defines in sqlutil.h, and is one of: <ul style="list-style-type: none"> • DMS • SMS |
| TBSP_CONTENT_TYPE | VARCHAR(10) | tablespace_content_type - Table space contents type. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • ANY • LARGE • SYSTEMP • USRTEMP |
| TBSP_PAGE_SIZE | BIGINT | tablespace_page_size - Table space page size |
| TBSP_EXTENT_SIZE | BIGINT | tablespace_extent_size - Table space extent size |
| TBSP_PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table space prefetch size |
| TBSP_CUR_POOL_ID | BIGINT | tablespace_cur_pool_id - Buffer pool currently being used |
| TBSP_NEXT_POOL_ID | BIGINT | tablespace_next_pool_id - Buffer pool that will be used at next startup |
| FS_CACHING | SMALLINT | fs_caching - File system caching |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |

Table 308. Information returned by the SNAP_GET_TBSP_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|---|
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer pool asynchronous read time |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests |
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer pool no victim buffers |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |

Table 308. Information returned by the SNAP_GET_TBSP_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|-------------|--|
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| FILES_CLOSED | BIGINT | files_closed - Database files closed |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| TBSP_REBALANCER_MODE | VARCHAR(10) | tablespace_rebalancer_mode - Rebalancer mode. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • NO_REBAL • FWD_REBAL • REV_REBAL |
| TBSP_USING_AUTO_STORAGE | SMALLINT | tablespace_using_auto_storage - Table space enabled for automatic storage |
| TBSP_AUTO_RESIZE_ENABLED | SMALLINT | tablespace_auto_resize_enabled - Table space automatic resizing enabled |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

SNAPTBSP QUIESCER administrative view and SNAP_GET_TBSP QUIESCER table function - Retrieve quiescer table space snapshot information

The SNAPTBSP QUIESCER administrative view and the SNAP_GET_TBSP QUIESCER table function return information about quiescers from a table space snapshot.

Important: The SNAPTBSP QUIESCER administrative view and SNAP_GET_TBSP QUIESCER table function are deprecated and have been replaced by “MON_GET_TABLESPACE QUIESCER table function - Get information about quiesced table spaces” on page 817.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPTBSP QUIESCER administrative view” on page 1420
- “SNAP_GET_TBSP QUIESCER table function” on page 1422

SNAPTbsp_QUIESCER administrative view

This administrative view allows you to retrieve quiescer table space snapshot information for the currently connected database.

Used with the SNAPTbsp_RANGE administrative view and the MON_GET_TABLESPACE and MON_GET_CONTAINER table functions, the SNAPTbsp_QUIESCER administrative view provides information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 309 on page 1423 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPTbsp_QUIESCER administrative view
- CONTROL privilege on the SNAPTbsp_QUIESCER administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_TBSP_QUIESCER table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve information about the quiesced table spaces for all database members for the currently connected database.

```
SELECT SUBSTR(TBSP_NAME, 1, 10) AS TBSP_NAME, QUIESCER_TS_ID,  
       QUIESCER_OBJ_ID, QUIESCER_AUTH_ID, QUIESCER_AGENT_ID,  
       QUIESCER_STATE, DBPARTITIONNUM  
FROM SYSIBMADM.SNAPTbsp_QUIESCER ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| TBSP_NAME | QUIESCER_TS_ID | QUIESCER_OBJ_ID | QUIESCER_AUTH_ID | .. |
|------------|----------------|-----------------|------------------|----|
| USERSPACE1 | 2 | 5 | SWALKTY | .. |
| USERSPACE1 | 2 | 5 | SWALKTY | .. |

2 record(s) selected.

Output from this query (continued).

| QUIESCER_AGENT_ID | QUIESCER_STATE | DBPARTITIONNUM |
|-------------------|----------------|----------------|
| 0 | EXCLUSIVE | 0 |
| 65983 | EXCLUSIVE | 1 |

Example: Determine the partitioned table names

If the table is partitioned and kept in quiesced state, the different values for table space ID and table ID are represented than in SYSCAT.TABLES. These IDs will appear as the unsigned short representation. In order to find the quiesced table name, you need to find the signed short representation first by calculating the table space ID that is subtracting 65536 (the maximum value) from QUIESCER_TS_ID and then use this table space ID to locate the quiesced tables. (The actual table space ID can be found in SYSCAT.DATAPARTITIONS for each data partition in the table.)

```
SELECT SUBSTR(TBSP_NAME, 1, 10) AS TBSP_NAME,
CASE WHEN QUIESCER_TS_ID = 65530
THEN QUIESCER_TS_ID - 65536
ELSE QUIESCER_TS_ID END as tbspaceid,
CASE WHEN QUIESCER_TS_ID = 65530
THEN QUIESCER_OBJ_ID - 65536
ELSE QUIESCER_OBJ_ID END as tableid
FROM SYSIBMADM.SNAPTbsp_QUIESCER
ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| TBSP_NAME | TBSPACEID | TABLEID |
|-----------|-----------|---------|
| TABDATA | -6 | -32768 |
| DATAMART | -6 | -32765 |
| SMALL | 5 | 17 |

3 record(s) selected.

Use the given TBSPACEID and TABLEID provided from this query to find the table schema and name from SYSCAT.TABLES.

```
SELECT CHAR(tabschema, 10) tabschema, CHAR(tabname, 15) tabname
FROM SYSCAT.TABLES
WHERE tbspaceid = -6 AND tableid in (-32768, -32765)
```

The following is an example of output from this query.

| TABSCHEMA | TABNAME |
|-----------|--------------|
| SAMP | ORDERS_RP |
| SAMP | ORDERS_DMART |

2 record(s) selected.

```
SELECT CHAR(tabschema, 10) tabschema, CHAR(tabname, 15) tabname
FROM SYSCAT.TABLES
WHERE tbspaceid = 5 AND tableid = 17
```

The following is an example of output from this query.

| TABSCHEMA | TABNAME |
|-----------|---------|
| SAMP | NATION |

1 record(s) selected.

SNAP_GET_TBSP QUIESCER table function

The SNAP_GET_TBSP QUIESCER table function returns the same information as the SNAPT BSP QUIESCER administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_TBSP, SNAP_GET_TBSP PART, SNAP_GET_TBSP RANGE, SNAP_GET_CONTAINER table functions, the SNAP_GET_TBSP QUIESCER table function provides information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 309 on page 1423 for a complete list of information that can be returned.

Syntax

```

>> SNAP_GET_TBSP QUIESCER (—dbname—)
                                |
                                |, member—
                                |
                                >>

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TBSP QUIESCER table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_TBSP QUIESCER table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve information about the quiesced table spaces for database member 1 for the currently connected database.

```
SELECT SUBSTR(TBSP_NAME, 1, 10) AS TBSP_NAME, QUIESCER_TS_ID,  
       QUIESCER_OBJ_ID, QUIESCER_AUTH_ID, QUIESCER_AGENT_ID,  
       QUIESCER_STATE, DBPARTITIONNUM  
FROM TABLE( SYSPROC.SNAP_GET_TBSP QUIESCER( '', 1)) AS T
```

The following is an example of output from this query.

| TBSP_NAME | QUIESCER_TS_ID | QUIESCER_OBJ_ID | QUIESCER_AUTH_ID | ... |
|------------|----------------|-----------------|------------------|-----|
| ----- | ----- | ----- | ----- | ... |
| USERSPACE1 | 2 | 5 | SWALKTY | ... |

1 record(s) selected.

Output from this query (continued).

| ... | QUIESCER_AGENT_ID | QUIESCER_STATE | DBPARTITIONNUM |
|-----|-------------------|----------------|----------------|
| ... | ----- | ----- | ----- |
| ... | 65983 | EXCLUSIVE | 1 |

Information returned

Table 309. Information returned by the SNAPTbsp_QUIESCER administrative view and the SNAP_GET_TBSP_QUIESCER table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| QUIESCER_TS_ID | BIGINT | quiescer_ts_id - Quiescer table space identification |
| QUIESCER_OBJ_ID | BIGINT | quiescer_obj_id - Quiescer object identification |

Table 309. Information returned by the `SNAPTbsp_QUIESCER` administrative view and the `SNAP_GET_TBSP_QUIESCER` table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------|--------------|--|
| QUIESCER_AUTH_ID | VARCHAR(128) | quiescer_auth_id - Quiescer user authorization identification |
| QUIESCER_AGENT_ID | BIGINT | quiescer_agent_id - Quiescer agent identification |
| QUIESCER_STATE | VARCHAR(14) | quiescer_state - Quiescer state. This interface returns a text identifier based on defines in <code>sqlutil.h</code> and is one of: <ul style="list-style-type: none"> • EXCLUSIVE • UPDATE • SHARE |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPUTIL administrative view and SNAP_GET_UTIL table function - Retrieve utility_info logical data group snapshot information

The `SNAPUTIL` administrative view and the `SNAP_GET_UTIL` table function return snapshot information about the utilities from the `utility_info` logical data group.

Important: The `SNAPUTIL` administrative view and the `SNAP_GET_UTIL` table function are deprecated and have been replaced by the “`MON_GET_UTILITY` table function - Get utilities running on the database” on page 863.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “`SNAPUTIL` administrative view”
- “`SNAP_GET_UTIL` table function” on page 1426

SNAPUTIL administrative view

Used in conjunction with the `SNAPUTIL_PROGRESS` administrative view, the `SNAPUTIL` administrative view provides the same information as the **LIST UTILITIES SHOW DETAIL** CLP command.

The schema is `SYSIBMADM`.

Refer to Table 310 on page 1427 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- `SELECT` privilege on the `SNAPUTIL` administrative view

- CONTROL privilege on the SNAPUTIL administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_UTIL table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of utilities and their states on all database members for all active databases in the instance that contains the connected database.

```
SELECT UTILITY_TYPE, UTILITY_PRIORITY, SUBSTR(UTILITY_DESCRIPTION, 1, 72)
      AS UTILITY_DESCRIPTION, SUBSTR(UTILITY_DBNAME, 1, 17) AS
      UTILITY_DBNAME, UTILITY_STATE, UTILITY_INVOKER_TYPE, DBPARTITIONNUM
FROM SYSIBMADM.SNAPUTIL ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| UTILITY_TYPE | UTILITY_PRIORITY | ... |
|--------------|------------------|-----|
| LOAD | - | ... |
| LOAD | - | ... |
| LOAD | - | ... |

3 record(s) selected.

Output from this query (continued).

| UTILITY_DESCRIPTION | ... |
|--|-----|
| ONLINE LOAD DEL AUTOMATIC INDEXING INSERT COPY NO TEST .LOADTEST | ... |
| ONLINE LOAD DEL AUTOMATIC INDEXING INSERT COPY NO TEST .LOADTEST | ... |
| ONLINE LOAD DEL AUTOMATIC INDEXING INSERT COPY NO TEST .LOADTEST | ... |

Output from this query (continued).

| UTILITY_DBNAME | UTILITY_STATE | UTILITY_INVOKER_TYPE | DBPARTITIONNUM |
|----------------|---------------|----------------------|----------------|
| SAMPLE | EXECUTE | USER | 0 |
| SAMPLE | EXECUTE | USER | 1 |
| SAMPLE | EXECUTE | USER | 2 |

SNAP_GET_UTIL table function

The SNAP_GET_UTIL table function returns the same information as the SNAPUTIL administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

Used in conjunction with the SNAP_GET_UTIL_PROGRESS table function, the SNAP_GET_UTIL table function provides the same information as the **LIST UTILITIES SHOW DETAIL** CLP command.

Refer to Table 310 on page 1427 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_UTIL (member) ►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_UTIL table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_UTIL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Retrieve a list of utility ids with their type and state for the currently connected database member on database SAMPLE.

```
SELECT UTILITY_ID, UTILITY_TYPE, UTILITY_STATE
FROM TABLE(SNAP_GET_UTIL(-1)) AS T WHERE UTILITY_DBNAME='SAMPLE'
```

The following is an example of output from this query:

| UTILITY_ID | UTILITY_TYPE | STATE |
|------------|--------------|---------|
| 1 | BACKUP | EXECUTE |

1 record(s) selected.

Example 2: Retrieve a list of utility ids with their type, member number and database partition number for the currently connected database member.

```
SELECT UTILITY_ID, UTILITY_TYPE, MEMBER, DBPARTITIONNUM
FROM TABLE(SNAP_GET_UTIL(-1)) AS T
```

The following is an example of output from this query:

| UTILITY_ID | UTILITY_TYPE | MEMBER | DBPARTITIONNUM |
|------------|--------------|--------|----------------|
| 2 | BACKUP | 2 | 2 |

Information returned

Table 310. Information returned by the SNAPUTIL administrative view and the SNAP_GET_UTIL table function

| Column name | Data type | Description or corresponding monitor element |
|---------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| UTILITY_ID | INTEGER | utility_id - Utility ID. Unique to a database partition. |
| UTILITY_TYPE | VARCHAR(26) | utility_type - Utility type. This interface returns a text identifier based on the defines in sqlmon.h. |
| UTILITY_PRIORITY | INTEGER | utility_priority - Utility priority. Priority if utility supports throttling, otherwise null. |
| UTILITY_DESCRIPTION | VARCHAR(2048) | utility_description - Utility description. Can be null. |
| UTILITY_DBNAME | VARCHAR(128) | utility_dbname - Database operated on by utility |
| UTILITY_START_TIME | TIMESTAMP | utility_start_time - Utility start time |
| UTILITY_STATE | VARCHAR(10) | utility_state - Utility state. This interface returns a text identifier based on the defines in sqlmon.h. |

Table 310. Information returned by the SNAPUTIL administrative view and the SNAP_GET_UTIL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|--|
| UTILITY_INVOKER_TYPE | VARCHAR(10) | utility_invoker_type - Utility invoker type. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> AUTO USER |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| PROGRESS_LIST_ATTR | VARCHAR(10) | progress_list_attr - Current progress list attributes |
| PROGRESS_LIST_CUR_SEQ_NUM | INTEGER | progress_list_current_seq_num - Current progress list sequence number |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAP_WRITE_FILE procedure

The SNAP_WRITE_FILE procedure writes system snapshot data to a file in the tmp subdirectory of the instance directory.

Important: The SNAP_WRITE_FILE procedure is deprecated and might be removed in a future release. Start using the monitoring SQL interfaces. If you still need to write snapshot data to a file, use other methods before this procedure becomes discontinued.

Syntax

►► SNAP_WRITE_FILE (—requestType—, —dbname—, —member—) ◀◀

The schema is SYSPROC.

Procedure parameters

requestType

An input argument of type VARCHAR (32) that specifies a valid snapshot request type. The possible request types are text identifiers based on defines in sqlmon.h, and are one of:

- APPL_ALL
- BUFFERPOOLS_ALL
- DB2
- DBASE_ALL
- DBASE_LOCKS
- DBASE_TABLES
- DBASE_TABLESPACES
- DYNAMIC_SQL

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database when calling this function. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An input argument of type INTEGER that specifies a valid member number. Specify -1 for the current member, or -2 for an aggregate of all active members. An active member is a member where the database is available for connection and use by applications.

If a null value is specified, -1 is set implicitly.

Authorization

One of the following authorizations is required to execute the procedure:

- EXECUTE privilege on the procedure
- DATAACCESS authority

To access snapshot monitor data, one of the following authorities is also required:

- SYSMON authority
- SYSCTRL authority
- SYSMAINT authority
- SYSADM authority

Note: If you do not have SYSMON, SYSCTRL, SYSMAINT, or SYSADM authority, you can read the saved snapshot data by passing null values as the inputs to snapshot functions.

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

Example

Take a snapshot of database manager information by specifying a request type of 'DB2' (which corresponds to SQLMA_DB2), and defaulting to the currently connected database and current database partition.

```
CALL SYSPROC.SNAP_WRITE_FILE ('DB2', '', -1)
```

This will result in snapshot data being written to the instance temporary directory, which is sqllib/tmp/SQLMA_DB2.dat on UNIX operating systems, and sqllib\DB2\tmp\SQLMA_DB2.dat on a Windows operating system.

Usage notes

If an unrecognized input parameter is provided, the following error is returned:
SQL2032N The "REQUEST_TYPE" parameter is not valid.

SNAPAGENT administrative view and SNAP_GET_AGENT table function - Retrieve agent logical data group application snapshot information

The SNAPAGENT administrative view and the SNAP_GET_AGENT table function return information about agents from an application snapshot, in particular, the agent logical data group.

Important: The SNAPAGENT administrative view and the SNAP_GET_AGENT table function are deprecated and have been replaced by the “MON_GET_AGENT table function - List agents, fenced mode processes, and system entities for the database” on page 551 and “MON_GET_CONNECTION table function - Get connection metrics” on page 588.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPAGENT administrative view” on page 1250
- “SNAP_GET_AGENT table function” on page 1251

SNAPAGENT administrative view

This administrative view allows you to retrieve agent logical data group application snapshot information for the currently connected database.

Used with the MON_GET_CONNECTION, MON_GET_MEMORY_POOL, and MON_GET_MEMORY_SET table functions and the MON_CURRENT_SQL and SNAPSUBSECTION administrative views, the SNAPAGENT administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS ON database-alias** CLP command, but retrieves data from all database members.

The schema is SYSIBMADM.

Refer to Table 274 on page 1253 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPAGENT administrative view
- CONTROL privilege on the SNAPAGENT administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_AGENT table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON

- SYSCTRL
- SYSMaint
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve all application snapshot information for the currently connected database from the agent logical data group.

```
SELECT * FROM SYSIBMADM.SNAPAGENT
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | DB_NAME | AGENT_ID | ... |
|----------------------------|---------|----------|-----|
| 2005-07-19-11.03.26.740423 | SAMPLE | 101 | ... |
| 2005-07-19-11.03.26.740423 | SAMPLE | 49 | ... |
| | | | ... |

2 record(s) selected.

Output from this query (continued).

| ... | AGENT_PID | LOCK_TIMEOUT_VAL | DBPARTITIONNUM |
|-----|-----------|------------------|----------------|
| ... | 11980 | -1 | 0 |
| ... | 15940 | -1 | 0 |
| ... | | | |
| ... | | | |

SNAP_GET_AGENT table function

The SNAP_GET_AGENT table function returns the same information as the SNAPAGENT administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_AGENT_MEMORY_POOL, SNAP_GET_APPL, SNAP_GET_APPL_INFO, SNAP_GET_STMT and SNAP_GET_SUBSECTION table functions, the SNAP_GET_AGENT table function provides information equivalent to the **GET SNAPSHOT FOR ALL APPLICATIONS** CLP command, but retrieves data from all database members.

Refer to Table 274 on page 1253 for a complete list of information that can be returned.

Syntax

```

--SNAP_GET_AGENT--(--dbname--  

└──, member──┘)

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_AGENT table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_AGENT table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve all application snapshot information for all applications in all active databases.

```
SELECT * FROM TABLE(SNAP_GET_AGENT(CAST(NULL AS VARCHAR(128))), -1)) AS T
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | DB_NAME | AGENT_ID | ... |
|----------------------------|---------|----------|-----|
| 2006-01-03-17.21.38.530785 | SAMPLE | 48 | ... |
| 2006-01-03-17.21.38.530785 | SAMPLE | 47 | ... |

```

2006-01-03-17.21.38.530785 SAMPLE          46 ...
2006-01-03-17.21.38.530785 TESTDB         30 ...
2006-01-03-17.21.38.530785 TESTDB         29 ...
2006-01-03-17.21.38.530785 TESTDB         28 ...

```

6 record(s) selected.

Output from this query (continued).

```

... AGENT_PID      LOCK_TIMEOUT_VAL  DBPARTITIONNUM
... -----
...           7696             -1             0
...           8536             -1             0
...           6672             -1             0
...           2332             -1             0
...           8360             -1             0
...           6736             -1             0
...

```

Information returned

Table 311. Information returned by the SNAPAGENT administrative view and the SNAP_GET_AGENT table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| AGENT_PID | BIGINT | agent_pid - Engine dispatchable unit (EDU) |
| LOCK_TIMEOUT_VAL | BIGINT | lock_timeout_val - Lock timeout (seconds) |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function - Retrieve memory_pool logical data group snapshot information

The SNAPAGENT_MEMORY_POOL administrative view and the SNAP_GET_AGENT_MEMORY_POOL table function return information about memory usage at the agent level.

Important: The SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function have been deprecated and replaced by the “MON_GET_MEMORY_POOL - get memory pool information” on page 679 and “MON_GET_MEMORY_SET - get memory set information” on page 682.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPAGENT_MEMORY_POOL administrative view” on page 1254

- “SNAP_GET_AGENT_MEMORY_POOL table function” on page 1255

SNAPAGENT_MEMORY_POOL administrative view

This administrative view allows you to retrieve the memory_pool logical data group snapshot information about memory usage at the agent level for the currently connected database.

Used with the MON_GET_AGENT and MON_GET_CONNECTION table functions and the MON_CURRENT_SQL and SNAPSUBSECTION administrative views, the SNAPAGENT_MEMORY_POOL administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 275 on page 1257 for a complete list of information that can be returned.

Authorization

One of the following is required to use the view:

- SELECT privilege on the SNAPAGENT_MEMORY_POOL administrative view
- CONTROL privilege on the SNAPAGENT_MEMORY_POOL administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_AGENT_MEMORY_POOL table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCtrl
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of memory pools and their current size.

```
SELECT AGENT_ID, POOL_ID, POOL_CUR_SIZE FROM SYSIBMADM.SNAPAGENT_MEMORY_POOL
```

The following is an example of output from this query.

| AGENT_ID | POOL_ID | POOL_ | CUR_SIZE |
|----------|--------------|-------|----------|
| 48 | APPLICATION | | 65536 |
| 48 | OTHER | | 65536 |
| 48 | APPL_CONTROL | | 65536 |
| 47 | APPLICATION | | 65536 |
| 47 | OTHER | | 131072 |
| 47 | APPL_CONTROL | | 65536 |
| 46 | OTHER | | 327680 |
| 46 | APPLICATION | | 262144 |
| 46 | APPL_CONTROL | | 65536 |

9 record(s) selected.

SNAP_GET_AGENT_MEMORY_POOL table function

The SNAP_GET_AGENT_MEMORY_POOL table function returns the same information as the SNAPAGENT_MEMORY_POOL administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_AGENT, SNAP_GET_APPL, SNAP_GET_APPL_INFO, SNAP_GET_STMT and SNAP_GET_SUBSECTION table functions, the SNAP_GET_AGENT_MEMORY_POOL table function provides information equivalent to the **GET SNAPSHOT FOR ALL APPLICATIONS** CLP command.

Refer to Table 275 on page 1257 for a complete list of information that can be returned.

Syntax

```

▶▶ SNAP_GET_AGENT_MEMORY_POOL ( (—dbname— [ , member— ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have

been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_AGENT_MEMORY_POOL table function takes a snapshot for the currently connected database and database member number.

Authorization

- One of the following authorizations is required:
- EXECUTE privilege on the SNAP_GET_AGENT_MEMORY_POOL table function
 - DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of memory pools and their current size for all databases.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AGENT_ID, POOL_ID, POOL_CUR_SIZE
FROM TABLE(SNAP_GET_AGENT_MEMORY_POOL(CAST (NULL AS VARCHAR(128)), -1))
AS T
```

The following is an example of output from this query.

| DB_NAME | AGENT_ID | POOL_ID | POOL_CUR_SIZE |
|---------|----------|--------------|---------------|
| SAMPLE | 48 | APPLICATION | 65536 |
| SAMPLE | 48 | OTHER | 65536 |
| SAMPLE | 48 | APPL_CONTROL | 65536 |
| SAMPLE | 47 | APPLICATION | 65536 |
| SAMPLE | 47 | OTHER | 131072 |
| SAMPLE | 47 | APPL_CONTROL | 65536 |
| SAMPLE | 46 | OTHER | 327680 |
| SAMPLE | 46 | APPLICATION | 262144 |
| SAMPLE | 46 | APPL_CONTROL | 65536 |
| TESTDB | 30 | APPLICATION | 65536 |
| TESTDB | 30 | OTHER | 65536 |
| TESTDB | 30 | APPL_CONTROL | 65536 |
| TESTDB | 29 | APPLICATION | 65536 |
| TESTDB | 29 | OTHER | 131072 |
| TESTDB | 29 | APPL_CONTROL | 65536 |
| TESTDB | 28 | OTHER | 327680 |
| TESTDB | 28 | APPLICATION | 65536 |
| TESTDB | 28 | APPL_CONTROL | 65536 |

18 record(s) selected.

Information returned

Table 312. Information returned by the SNAPAGENT_MEMORY_POOL administrative view and the SNAP_GET_AGENT_MEMORY_POOL table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| AGENT_PID | BIGINT | agent_pid - Engine dispatchable unit (EDU) |
| POOL_ID | VARCHAR(14) | pool_id - Memory pool identifier. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • APP_GROUP • APPL_CONTROL • APPLICATION • BP • CAT_CACHE • DATABASE • DFM • FCMBP • IMPORT_POOL • LOCK_MGR • MONITOR • OTHER • PACKAGE_CACHE • QUERY • SHARED_SORT • SORT • STATEMENT • STATISTICS • UTILITY |
| POOL_CUR_SIZE | BIGINT | pool_cur_size - Current size of memory pool |
| POOL_WATERMARK | BIGINT | pool_watermark - Memory pool watermark |
| POOL_CONFIG_SIZE | BIGINT | pool_config_size - Configured size of memory pool |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function - Retrieve appl_info logical data group snapshot information

The SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function return information about applications from an application snapshot, in particular, the appl_info logical data group.

Important: The SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function are deprecated and have been replaced by the “MON_GET_CONNECTION table function - Get connection metrics” on page 588.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPAPPL_INFO administrative view” on page 1258
- “SNAP_GET_APPL_INFO table function” on page 1259

SNAPAPPL_INFO administrative view

This administrative view allows you to retrieve appl_info logical data group snapshot information for the currently connected database.

Used with the MON_GET_AGENT, MON_GET_CONNECTION, MON_GET_MEMORY_POOL, and MON_GET_MEMORY_SET table functions and the MON_CURRENT_SQL and SNAPSUBSECTION administrative views, the SNAPAPPL_INFO administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS ON database-alias** CLP command, but retrieves data from all database members.

The schema is SYSIBMADM.

Refer to Table 276 on page 1261 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPAPPL_INFO administrative view
- CONTROL privilege on the SNAPAPPL_INFO administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_APPL_INFO table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Also, one of the following authorities is required:

- SYSMON
- SYSMANT

- SYSCTRL
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve the status of the applications connected to the current database.

```
SELECT AGENT_ID, SUBSTR(APPL_NAME,1,10) AS APPL_NAME, APPL_STATUS
FROM SYSIBMADM.SNAPAPPL_INFO
```

The following is an example of output from this query.

| AGENT_ID | APPL_NAME | APPL_STATUS |
|----------|-----------|-------------|
| 101 | db2bp.exe | UOWEXEC |
| 49 | db2bp.exe | CONNECTED |

2 record(s) selected.

SNAP_GET_APPL_INFO table function

The SNAP_GET_APPL_INFO table function returns the same information as the SNAPAPPL_INFO administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_AGENT, SNAP_GET_AGENT_MEMORY_POOL, SNAP_GET_APPL, SNAP_GET_STMT and SNAP_GET_SUBSECTION table functions, the SNAP_GET_APPL_INFO table function provides information equivalent to the **GET SNAPSHOT FOR ALL APPLICATIONS** CLP command, but retrieves data from all database members.

Refer to Table 276 on page 1261 for a complete list of information that can be returned.

Syntax

```
➤➤ SNAP_GET_APPL_INFO ( (dbname [ , member ] ) ) ➤➤
```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_APPL_INFO table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_APPL_INFO table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Retrieve the status of all applications on the connected database member.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AGENT_ID,  
       SUBSTR(APPL_NAME,1,10) AS APPL_NAME, APPL_STATUS  
FROM TABLE(SNAP_GET_APPL_INFO(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

| DB_NAME | AGENT_ID | APPL_NAME | APPL_STATUS |
|---------|----------|-----------|-------------|
| TOOLSDB | 14 | db2bp.exe | CONNECTED |
| SAMPLE | 15 | db2bp.exe | UOWEXEC |
| SAMPLE | 8 | javaw.exe | CONNECTED |
| SAMPLE | 7 | db2bp.exe | UOWWAIT |

4 record(s) selected.

The following shows what you obtain when you SELECT from the result of the table function.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AUTHORITY_LVL  
FROM TABLE(SNAP_GET_APPL_INFO(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

```
DB_NAME  AUTHORITY_LVL
-----
TESTDB   SYSADM(GROUP) + DBADM(USER) + CREATETAB(USER, GROUP) +
          BINDADD(USER, GROUP) + CONNECT(USER, GROUP) +
          CREATE_NOT_FENC(USER) + IMPLICIT_SCHEMA(USER, GROUP) +
          LOAD(USER) + CREATE_EXT_RT(USER) + QUIESCE_CONN(USER)
TESTDB   SYSADM(GROUP) + DBADM(USER) + CREATETAB(USER, GROUP) +
          BINDADD(USER, GROUP) + CONNECT(USER, GROUP) +
          CREATE_NOT_FENC(USER) + IMPLICIT_SCHEMA(USER, GROUP) +
          LOAD(USER) + CREATE_EXT_RT(USER) + QUIESCE_CONN(USER)
TESTDB   SYSADM(GROUP) + DBADM(USER) + CREATETAB(USER, GROUP) +
          BINDADD(USER, GROUP) + CONNECT(USER, GROUP) +
          CREATE_NOT_FENC(USER) + IMPLICIT_SCHEMA(USER, GROUP) +
          LOAD(USER) + CREATE_EXT_RT(USER) + QUIESCE_CONN(USER)
```

3 record(s) selected.

Information returned

Table 313. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |

Table 313. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------|-------------|--|
| APPL_STATUS | VARCHAR(22) | <p>appl_status - Application status. This interface returns a text identifier based on the defines in sqlmon.h, and is one of:</p> <ul style="list-style-type: none"> • BACKUP • COMMIT_ACT • COMP • CONNECTED • CONNECTPEND • CREATE_DB • DECOUPLED • DISCONNECTPEND • INTR • IOERROR_WAIT • LOAD • LOCKWAIT • QUIESCE_TABLESPACE • RECOMP • REMOTE_RQST • RESTART • RESTORE • ROLLBACK_ACT • ROLLBACK_TO_SAVEPOINT • TEND • THABRT • THCOMT • TPREP • UNLOAD • UOWEXEC • UOWWAIT • WAITFOR_REMOTE |
| CODEPAGE_ID | BIGINT | codepage_id - ID of code page used by application |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of associated agents |
| COORD_NODE_NUM | SMALLINT | COORD_NODE_NUM is deprecated and is replaced by MEMBER. |

Table 313. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------|--------------|--|
| AUTHORITY_LVL | VARCHAR(512) | <p>authority_bitmap - User authorization level.</p> <p>This interface returns a text identifier based on the database authorities defined in sql.h and their source, and has the following format: authority(source, ...) + authority(source, ...) + ... The source of an authority can be multiple: either from a USER, a GROUP, or a USER and a GROUP.</p> <p>Possible values for "authority":</p> <ul style="list-style-type: none"> • ACCESSCTRL • BINDADD • CONNECT • CREATE_EXT_RT • CREATE_NOT_FENC • CREATETAB • DATAACCESS • DBADM • EXPLAIN • IMPLICIT_SCHEMA • LOAD • LIBADM • QUIESCE_CONN • SECADM • SQLADM • SYSADM • SYSCTRL • SYSMANT • SYSMON • SYSQUIESCE • WLMADM <p>Possible values for "source":</p> <ul style="list-style-type: none"> • USER - authority granted to the user or to a role granted to the user. • GROUP - authority granted to a group to which the user belongs or to a role granted to the group to which the user belongs. |
| CLIENT_PID | BIGINT | client_pid - Client process ID |
| COORD_AGENT_PID | BIGINT | coord_agent_pid - Coordinator agent |

Table 313. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|--|
| STATUS_CHANGE_TIME | TIMESTAMP | status_change_time - Application status change time |
| CLIENT_PLATFORM | VARCHAR(12) | <p>client_platform - Client operating platform. This interface returns a text identifier based on the defines in sqlmon.h,</p> <ul style="list-style-type: none"> • AIX • AIX64 • AS400_DRDA • DOS • DYNIX • HP • HP64 • HPIA • HPIA64 • LINUX • LINUX390 • LINUXIA64 • LINUXPPC • LINUXPPC64 • LINUXX8664 • LINUXZ64 • MAC • MVS_DRDA • NT • NT64 • OS2 • OS390 • SCO • SGI • SNI • SUN • SUN64 • UNKNOWN • UNKNOWN_DRDA • VM_DRDA • VSE_DRDA • WINDOWS |

Table 313. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|---------------|---|
| CLIENT_PROTOCOL | VARCHAR(10) | client_protocol - Client communication protocol. This interface returns a text identifier based on the defines in sqlmon.h, <ul style="list-style-type: none"> • CPIC • LOCAL • NPIPE • TCPIP (for Db2) • TCPIP4 • TCPIP6 |
| TERRITORY_CODE | SMALLINT | territory_code - Database territory code |
| APPL_NAME | VARCHAR(256) | appl_name - Application name |
| APPL_ID | VARCHAR(128) | appl_id - Application ID |
| SEQUENCE_NO | VARCHAR(4) | sequence_no - Sequence number |
| PRIMARY_AUTH_ID | VARCHAR(128) | auth_id - Authorization ID |
| SESSION_AUTH_ID | VARCHAR(128) | session_auth_id - Session authorization ID |
| CLIENT_NNAME | VARCHAR(128) | client_nname - Client name monitor element |
| CLIENT_PRDID | VARCHAR(128) | client_prdid - Client product/version ID |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input database alias |
| CLIENT_DB_ALIAS | VARCHAR(128) | client_db_alias - Database alias used by application |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| DB_PATH | VARCHAR(1024) | db_path - Database path |
| EXECUTION_ID | VARCHAR(128) | execution_id - User login ID |
| CORR_TOKEN | VARCHAR(128) | corr_token - DRDA correlation token |
| TPMON_CLIENT_USERID | VARCHAR(256) | tpmon_client_userid - TP monitor client user ID |
| TPMON_CLIENT_WKSTN | VARCHAR(256) | tpmon_client_wkstn - TP monitor client workstation name |
| TPMON_CLIENT_APP | VARCHAR(256) | tpmon_client_app - TP monitor client application name |
| TPMON_ACC_STR | VARCHAR(255) | tpmon_acc_str - TP monitor client accounting string |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| WORKLOAD_ID | INTEGER | workload_id - Workload ID monitor element |

Table 313. Information returned by the SNAPAPPL_INFO administrative view and the SNAP_GET_APPL_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|-----------|--|
| IS_SYSTEM_APPL | SMALLINT | is_system_appl - Is System Application monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| COORD_MEMBER | SMALLINT | coord_member - Coordinator member monitor element |
| COORD_DBPARTITIONNUM | SMALLINT | The coordinating database partition number. |

SNAPAPPL administrative view and SNAP_GET_APPL table function - Retrieve appl logical data group snapshot information

The SNAPAPPL administrative view and the SNAP_GET_APPL table function return information about applications from an application snapshot, in particular, the appl logical data group.

Important: The SNAPAPPL administrative view and the SNAP_GET_APPL table function are deprecated and have been replaced by the “MON_GET_CONNECTION table function - Get connection metrics” on page 588.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPAPPL administrative view” on page 1272
- “SNAP_GET_APPL table function” on page 1273

SNAPAPPL administrative view

This administrative view allows you to retrieve appl logical data group snapshot information for the currently connected database.

Used with the MON_GET_AGENT, MON_GET_CONNECTION, MON_GET_MEMORY_POOL, and MON_GET_MEMORY_SET table functions and the MON_CURRENT_SQL and SNAPSUBSECTION administrative views, the SNAPAPPL administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS ON database-alias** CLP command, but retrieves data from all database members.

The schema is SYSIBMADM.

Refer to Table 278 on page 1275 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPAPPL administrative view
- CONTROL privilege on the SNAPAPPL administrative view

- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following privileges or authorities is required to use the table function:

- EXECUTE privilege on the SNAP_GET_APPL table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve details on rows read and written for each application in the connected database.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AGENT_ID, ROWS_READ, ROWS_WRITTEN
FROM SYSIBMADM.SNAPAPPL
```

The following is an example of output from this query.

| DB_NAME | AGENT_ID | ROWS_READ | ROWS_WRITTEN |
|---------|----------|-----------|--------------|
| SAMPLE | | 7 | 25 |

1 record(s) selected.

SNAP_GET_APPL table function

The SNAP_GET_APPL table function returns the same information as the SNAPAPPL administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_AGENT, SNAP_GET_AGENT_MEMORY_POOL, SNAP_GET_APPL_INFO, SNAP_GET_STMT and SNAP_GET_SUBSECTION table functions, the SNAP_GET_APPL table function provides information equivalent to the **GET SNAPSHOT FOR ALL APPLICATIONS** CLP command, but retrieves data from all database members.

Refer to Table 278 on page 1275 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_APPL ((*dbname* [, *member*])) ►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_APPL table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_APPL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve details on rows read and written for each application for all active databases.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, AGENT_ID, ROWS_READ, ROWS_WRITTEN
FROM TABLE (SNAP_GET_APPL(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

| DB_NAME | AGENT_ID | ROWS_READ | ROWS_WRITTEN |
|---------|----------|-----------|--------------|
| WSDB | 679 | 0 | 0 |
| WSDB | 461 | 3 | 0 |
| WSDB | 460 | 4 | 0 |
| TEST | 680 | 4 | 0 |
| TEST | 455 | 6 | 0 |
| TEST | 454 | 0 | 0 |
| TEST | 453 | 50 | 0 |

Information returned

Table 314. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| UOW_LOG_SPACE_USED | BIGINT | uow_log_space_used - Unit of work log space used |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| INACT_STMTHIST_SZ | BIGINT | inact_stmthist_sz - Statement history list size |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |

Table 314. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| LOCKS_HELD | BIGINT | locks_held - Locks held |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive lock escalations |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |
| TOTAL_SORTS | BIGINT | total_sorts - Total sorts |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total sort time |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |

Table 314. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|-----------|---|
| COMMIT_SQL_STMTS | BIGINT | commit_sql_stmts - Commit statements attempted |
| ROLLBACK_SQL_STMTS | BIGINT | rollback_sql_stmts - Rollback statements attempted |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL statements attempted |
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL statements attempted |
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed statement operations |
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL statements executed |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data definition language (DDL) SQL statements |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - UPDATE/INSERT/DELETE SQL statements executed |
| INT_AUTO_REBINDS | BIGINT | int_auto_rebinds - Internal automatic rebinds |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal rows deleted |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal rows updated |
| INT_COMMITS | BIGINT | int_commits - Internal commits |
| INT_ROLLBACKS | BIGINT | int_rollback - Internal rollbacks |
| INT_DEADLOCK_ROLLBACKS | BIGINT | int_deadlock_rollback - Internal rollbacks due to deadlock |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated |
| ROWS_SELECTED | BIGINT | rows_selected - Rows selected |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/precompiles attempted |
| OPEN_REM_CURS | BIGINT | open_rem_curs - Open remote cursors |
| OPEN_REM_CURS_BLK | BIGINT | open_rem_curs_blk - Open remote cursors with blocking |
| REJ_CURS_BLK | BIGINT | rej_curs_blk - Rejected block cursor requests |
| ACC_CURS_BLK | BIGINT | acc_curs_blk - Accepted block cursor requests |
| SQL_REQS_SINCE_COMMIT | BIGINT | sql_reqs_since_commit - SQL requests since last commit |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |

Table 314. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|---|
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal rows inserted |
| OPEN_LOC_CURS | BIGINT | open_loc_curs - Open local cursors |
| OPEN_LOC_CURS_BLK | BIGINT | open_loc_curs_blk - Open local cursors with blocking |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog cache overflows |
| NUM_AGENTS | BIGINT | num_agents - Number of agents working on a statement |
| AGENTS_STOLEN | BIGINT | agents_stolen - Stolen agents |
| ASSOCIATED_AGENTS_TOP | BIGINT | associated_agents_top - Maximum number of associated agents |
| APPL_PRIORITY | BIGINT | appl_priority - Application agent priority |
| APPL_PRIORITY_TYPE | VARCHAR(16) | appl_priority_type - Application priority type. This interface returns a text identifier, based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • DYNAMIC_PRIORITY • FIXED_PRIORITY |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - Section lookups |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Section inserts |
| LOCKS_WAITING | BIGINT | locks_waiting - Current agents waiting on locks |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total hash joins |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total hash loops |
| HASH_JOIN_OVERFLOWS | BIGINT | hash_join_overflows - Hash join overflows |
| HASH_JOIN_SMALL_OVERFLOWS | BIGINT | hash_join_small_overflows - Hash join small overflows |
| APPL_IDLE_TIME | BIGINT | appl_idle_time - Application idle time |

Table 314. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|-------------|--|
| UOW_LOCK_WAIT_TIME | BIGINT | uow_lock_wait_time - Total time unit of work waited on locks |
| UOW_COMP_STATUS | VARCHAR(14) | uow_comp_status - Unit of work completion status. This interface returns a text identifier, based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • APPL_END • UOWABEND • UOWCOMMIT • UOWDEADLOCK • UOWLOCKTIMEOUT • UOWROLLBACK • UOWUNKNOWN |
| AGENT_USR_CPU_TIME_S | BIGINT | agent_usr_cpu_time - User CPU time used by agent (in seconds)* |
| AGENT_USR_CPU_TIME_MS | BIGINT | agent_usr_cpu_time - User CPU time used by agent (fractional, in microseconds)* |
| AGENT_SYS_CPU_TIME_S | BIGINT | agent_sys_cpu_time - System CPU time used by agent (in seconds)* |
| AGENT_SYS_CPU_TIME_MS | BIGINT | agent_sys_cpu_time - System CPU time used by agent (fractional, in microseconds)* |
| APPL_CON_TIME | TIMESTAMP | appl_con_time - Connection request start timestamp |
| CONN_COMPLETE_TIME | TIMESTAMP | conn_complete_time - Connection request completion timestamp |
| LAST_RESET | TIMESTAMP | last_reset - Last reset timestamp |
| UOW_START_TIME | TIMESTAMP | uow_start_time - Unit of work start timestamp |
| UOW_STOP_TIME | TIMESTAMP | uow_stop_time - Unit of work stop timestamp |
| PREV_UOW_STOP_TIME | TIMESTAMP | prev_uow_stop_time - Previous unit of work completion timestamp |
| UOW_ELAPSED_TIME_S | BIGINT | uow_elapsed_time - Most recent unit of work elapsed time (in seconds)* |
| UOW_ELAPSED_TIME_MS | BIGINT | uow_elapsed_time - Most recent unit of work elapsed time (fractional, in microseconds)* |
| ELAPSED_EXEC_TIME_S | BIGINT | elapsed_exec_time - Statement execution elapsed time (in seconds)* |
| ELAPSED_EXEC_TIME_MS | BIGINT | elapsed_exec_time - Statement execution elapsed time (fractional, in microseconds)* |

Table 314. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--|-------------|--|
| INBOUND_COMM_ADDRESS | VARCHAR(32) | inbound_comm_address - Inbound communication address |
| LOCK_TIMEOUT_VAL | BIGINT | lock_timeout_val - Lock timeout (seconds) |
| PRIV_WORKSPACE_NUM_OVERFLOWS | BIGINT | priv_workspace_num_overflows - Private workspace overflows |
| PRIV_WORKSPACE_SECTION_INSERTS | BIGINT | priv_workspace_section_inserts - Private workspace section inserts |
| PRIV_WORKSPACE_SECTION_LOOKUPS | BIGINT | priv_workspace_section_lookups - Private workspace section lookups |
| PRIV_WORKSPACE_SIZE_TOP | BIGINT | priv_workspace_size_top - Maximum private workspace size |
| SHR_WORKSPACE_NUM_OVERFLOWS | BIGINT | shr_workspace_num_overflows - Shared workspace overflows |
| SHR_WORKSPACE_SECTION_INSERTS | BIGINT | shr_workspace_section_inserts - Shared workspace section inserts |
| SHR_WORKSPACE_SECTION_LOOKUPS | BIGINT | shr_workspace_section_lookups - Shared workspace section lookups |
| SHR_WORKSPACE_SIZE_TOP | BIGINT | shr_workspace_size_top - Maximum shared workspace size |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| CAT_CACHE_SIZE_TOP | BIGINT | cat_cache_size_top - Catalog cache high water mark |
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP functions |
| OLAP_FUNC_OVERFLOWS | BIGINT | olap_func_overflows - OLAP function overflows |
| MEMBER | SMALLINT | member - Database member monitor element |
| <p>* To calculate the total time spent for the monitor element that this column is based on, you must add the full seconds reported in the column for this monitor element that ends with _S to the fractional seconds reported in the column for this monitor element that ends with _MS, using the following formula: $(\text{monitor-element-name_S} \times 1,000,000 + \text{monitor-element-name_MS}) \div 1,000,000$. For example, $(\text{ELAPSED_EXEC_TIME_S} \times 1,000,000 + \text{ELAPSED_EXEC_TIME_MS}) \div 1,000,000$.</p> | | |

SNAPBP administrative view and SNAP_GET_BP table function - Retrieve bufferpool logical group snapshot information

The SNAPBP administrative view and the SNAP_GET_BP table function return information about buffer pools from a bufferpool snapshot, in particular, the bufferpool logical data group.

Important: The SNAPBP administrative view and the SNAP_GET_BP table function are deprecated and have been replaced by the “MON_GET_BUFFERPOOL table function - Get buffer pool metrics” on page 571.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPBP administrative view” on page 1288
- “SNAP_GET_BP table function” on page 1289

SNAPBP administrative view

This administrative view allows you to retrieve bufferpool logical group snapshot information for the currently connected database.

Used with the MON_GET_BUFFERPOOL table function, the SNAPBP administrative view provides the data equivalent to the **GET SNAPSHOT FOR BUFFERPOOLS ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 280 on page 1290 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPBP administrative view
- CONTROL privilege on the SNAPBP administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_BP table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve data and index writes for all the bufferpools of the currently connected database.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME,SUBSTR(BP_NAME,1,15)
      AS BP_NAME,POOL_DATA_WRITES,POOL_INDEX_WRITES
FROM SYSIBMADM.SNAPBP
```

The following is an example of output from this query.

| DB_NAME | BP_NAME | POOL_DATA_WRITES | POOL_INDEX_WRITES |
|---------|----------------|------------------|-------------------|
| TEST | IBMDEFAULTBP | 0 | 0 |
| TEST | IBMSYSTEMBP4K | 0 | 0 |
| TEST | IBMSYSTEMBP8K | 0 | 0 |
| TEST | IBMSYSTEMBP16K | 0 | 0 |
| TEST | IBMSYSTEMBP32K | 0 | 0 |

5 record(s) selected

SNAP_GET_BP table function

The SNAP_GET_BP table function returns the same information as the SNAPBP administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_BP_PART table function, the SNAP_GET_BP table function provides the data equivalent to the **GET SNAPSHOT FOR ALL BUFFERPOOLS** CLP command.

Refer to Table 280 on page 1290 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_BP ((*dbname* [, *member*])) ►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database

members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_BP table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_BP table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve total physical and logical reads for all bufferpools for all active databases for the currently connected database member.

```
SELECT SUBSTR(T.DB_NAME,1,10) AS DB_NAME,
       SUBSTR(T.BP_NAME,1,20) AS BP_NAME,
       (T.POOL_DATA_L_READS+T.POOL_INDEX_L_READS) AS TOTAL_LOGICAL_READS,
       (T.POOL_DATA_P_READS+T.POOL_INDEX_P_READS) AS TOTAL_PHYSICAL_READS,
       T.DBPARTITIONNUM
FROM TABLE(SNAP_GET_BP(CAST(NULL AS VARCHAR(128)), -1)) AS T
```

The following is an example of output from this query.

| DB_NAME | BP_NAME | TOTAL_LOGICAL_READS | ... |
|---------|--------------|---------------------|-----|
| ----- | ----- | ----- | ... |
| SAMPLE | IBMDEFAULTBP | 0 | ... |
| TOOLSDB | IBMDEFAULTBP | 0 | ... |
| TOOLSDB | BP32K0000 | 0 | ... |

3 record(s) selected.

Output from this query (continued).

| ... | TOTAL_PHYSICAL_READS | DBPARTITIONNUM |
|-----|----------------------|----------------|
| ... | ----- | ----- |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |

Information returned

Table 315. Information returned by the SNAPBP administrative view and the SNAP_GET_BP table function

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|---------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| BP_NAME | VARCHAR(128) | bp_name - Buffer pool name |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| DB_PATH | VARCHAR(1024) | db_path - Database path |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input database alias |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes |

Table 315. Information returned by the SNAPBP administrative view and the SNAP_GET_BP table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|---|
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer pool asynchronous read time |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| FILES_CLOSED | BIGINT | files_closed - Database files closed |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer pool no victim buffers |
| PAGES_FROM_BLOCK_IOS | BIGINT | pages_from_block_ios - Total number of pages read by block I/O |

Table 315. Information returned by the SNAPBP administrative view and the SNAP_GET_BP table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|--|
| PAGES_FROM_VECTORED_IOS | BIGINT | pages_from_vectored_ios - Total pages read by vectored I/O |
| VECTORED_IOS | BIGINT | vectored_ios - Number of vectored I/O requests |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPBP_PART administrative view and SNAP_GET_BP_PART table function - Retrieve bufferpool_nodeinfo logical data group snapshot information

The SNAPBP_PART administrative view and the SNAP_GET_BP_PART table function return information about buffer pools from a bufferpool snapshot, in particular, the bufferpool_nodeinfo logical data group.

Important: The SNAPBP_PART administrative view and the SNAP_GET_BP_PART table function are deprecated and have been replaced by the “MON_GET_BUFFERPOOL table function - Get buffer pool metrics” on page 571.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPBP_PART administrative view” on page 1293
- “SNAP_GET_BP_PART table function” on page 1294

SNAPBP_PART administrative view

This administrative view allows you to retrieve bufferpool_nodeinfo logical data group snapshot information for the currently connected database.

Used with the MON_GET_BUFFERPOOL table function, the SNAPBP_PART administrative view provides the data equivalent to the **GET SNAPSHOT FOR BUFFERPOOLS ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 281 on page 1296 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPBP_PART administrative view
- CONTROL privilege on the SNAPBP_PART administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_BP_PART table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve data for all bufferpools when connected to SAMPLE database.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, SUBSTR(BP_NAME,1,15) AS BP_NAME,
       BP_CUR_BUFFSZ, BP_NEW_BUFFSZ, BP_PAGES_LEFT_TO_REMOVE, BP_TBSP_USE_COUNT
FROM SYSIBMADM.SNAPBP_PART
```

The following is an example of output from this query.

| DB_NAME | BP_NAME | BP_CUR_BUFFSZ | BP_NEW_BUFFSZ | ... |
|---------|----------------|---------------|---------------|-----|
| SAMPLE | IBMDEFAULTBP | 1000 | 1000 | ... |
| SAMPLE | IBMSYSTEMBP4K | 16 | 16 | ... |
| SAMPLE | IBMSYSTEMBP8K | 16 | 16 | ... |
| SAMPLE | IBMSYSTEMBP16K | 16 | 16 | ... |

4 record(s) selected.

Output from this query (continued).

| ... | BP_PAGES_LEFT_TO_REMOVE | BP_TBSP_USE_COUNT |
|-----|-------------------------|-------------------|
| ... | 0 | 3 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | | |

SNAP_GET_BP_PART table function

The SNAP_GET_BP_PART table function returns the same information as the SNAPBP_PART administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_BP table function, the SNAP_GET_BP_PART table function provides the data equivalent to the **GET SNAPSHOT FOR ALL BUFFERPOOLS** CLP command.

Refer to Table 281 on page 1296 for a complete list of information that can be returned.

Syntax

▶▶ SNAP_GET_BP_PART (—*dbname*—
 └─, *member*—) —————▶▶

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot for all bufferpools in all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_BP_PART table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_BP_PART table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve data for all bufferpools for all active databases when connected to the SAMPLE database.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, SUBSTR(BP_NAME,1,15) AS BP_NAME,
       BP_CUR_BUFFSZ, BP_NEW_BUFFSZ, BP_PAGES_LEFT_TO_REMOVE, BP_TBSP_USE_COUNT
FROM TABLE(SNAP_GET_BP_PART(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

| DB_NAME | BP_NAME | BP_CUR_BUFFSZ | BP_NEW_BUFFSZ | ... |
|---------|----------------|---------------|---------------|-----|
| SAMPLE | IBMDEFAULTBP | 250 | 250 | ... |
| SAMPLE | IBMSYSTEMBP4K | 16 | 16 | ... |
| SAMPLE | IBMSYSTEMBP8K | 16 | 16 | ... |
| SAMPLE | IBMSYSTEMBP16K | 16 | 16 | ... |
| SAMPLE | IBMSYSTEMBP32K | 16 | 16 | ... |
| TESTDB | IBMDEFAULTBP | 250 | 250 | ... |
| TESTDB | IBMSYSTEMBP4K | 16 | 16 | ... |
| TESTDB | IBMSYSTEMBP8K | 16 | 16 | ... |
| TESTDB | IBMSYSTEMBP16K | 16 | 16 | ... |
| TESTDB | IBMSYSTEMBP32K | 16 | 16 | ... |

...

Output from this query (continued).

| ... | BP_PAGES_LEFT_TO_REMOVE | BP_TBSP_USE_COUNT |
|-----|-------------------------|-------------------|
| ... | 0 | 3 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 3 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |
| ... | 0 | 0 |

...

Information returned

Table 316. Information returned by the SNAPBP_PART administrative view and the SNAP_GET_BP_PART table function

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| BP_NAME | VARCHAR(128) | bp_name - Buffer pool name |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| BP_CUR_BUFFSZ | BIGINT | bp_cur_buffsz - current size of buffer pool |
| BP_NEW_BUFFSZ | BIGINT | bp_new_buffsz - New buffer pool size |
| BP_PAGES_LEFT_TO_REMOVE | BIGINT | bp_pages_left_to_remove - Number of pages left to remove |
| BP_TBSP_USE_COUNT | BIGINT | bp_tbsp_use_count - Number of table spaces mapped to buffer pool |

Table 316. Information returned by the SNAPBP_PART administrative view and the SNAP_GET_BP_PART table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------|-----------|--|
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPCONTAINER administrative view and SNAP_GET_CONTAINER table function - Retrieve tablespace_container logical data group snapshot information

The SNAPCONTAINER administrative view and the SNAP_GET_CONTAINER table function return table space snapshot information from the tablespace_container logical data group.

Important: The SNAPCONTAINER administrative view and the SNAP_GET_CONTAINER table function are deprecated and have been replaced by the “MON_GET_CONTAINER table function - Get table space container metrics” on page 616.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPCONTAINER administrative view” on page 1301
- “SNAP_GET_CONTAINER table function” on page 1302

SNAPCONTAINER administrative view

This administrative view allows you to retrieve tablespace_container logical data group snapshot information for the currently connected database.

Used with the MON_GET_TABLESPACE table function and the SNAPTBSP_QUIESCER SNAPTBSP_RANGE administrative view, the SNAPCONTAINER administrative view returns data equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 283 on page 1304 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPCONTAINER administrative view
- CONTROL privilege on the SNAPCONTAINER administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_CONTAINER table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMOINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve details for the table space containers for all database members for the currently connected database.

```
SELECT SNAPSHOT_TIMESTAMP, SUBSTR(TBSP_NAME, 1, 15) AS TBSP_NAME,
       TBSP_ID, SUBSTR(CONTAINER_NAME, 1, 20) AS CONTAINER_NAME,
       CONTAINER_ID, CONTAINER_TYPE, ACCESSIBLE, DBPARTITIONNUM
FROM SYSIBMADM.SNAPCONTAINER ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | TBSP_NAME | TBSP_ID | ... |
|----------------------------|--------------|---------|-----|
| 2006-01-08-16.49.24.639945 | SYSCATSPACE | 0 | ... |
| 2006-01-08-16.49.24.639945 | TEMPSPACE1 | 1 | ... |
| 2006-01-08-16.49.24.639945 | USERSPACE1 | 2 | ... |
| 2006-01-08-16.49.24.639945 | SYSTOOLSPACE | 3 | ... |
| 2006-01-08-16.49.24.640747 | TEMPSPACE1 | 1 | ... |
| 2006-01-08-16.49.24.640747 | USERSPACE1 | 2 | ... |
| 2006-01-08-16.49.24.639981 | TEMPSPACE1 | 1 | ... |
| 2006-01-08-16.49.24.639981 | USERSPACE1 | 2 | ... |

8 record(s) selected.

Output from this query (continued).

| ... | CONTAINER_NAME | CONTAINER_ID | CONTAINER_TYPE | ... |
|-----|----------------------|--------------|-----------------|-----|
| ... | /home/swalkty/swalkt | 0 | FILE_EXTENT_TAG | ... |
| ... | /home/swalkty/swalkt | 0 | PATH | ... |
| ... | /home/swalkty/swalkt | 0 | FILE_EXTENT_TAG | ... |
| ... | /home/swalkty/swalkt | 0 | FILE_EXTENT_TAG | ... |
| ... | /home/swalkty/swalkt | 0 | PATH | ... |
| ... | /home/swalkty/swalkt | 0 | FILE_EXTENT_TAG | ... |
| ... | /home/swalkty/swalkt | 0 | PATH | ... |
| ... | /home/swalkty/swalkt | 0 | FILE_EXTENT_TAG | ... |

Output from this query (continued).

| ... | ACCESSIBLE | DBPARTITIONNUM |
|-----|------------|----------------|
| ... | 1 | 0 |
| ... | 1 | 0 |
| ... | 1 | 0 |

| | | |
|-----|---|---|
| ... | 1 | 0 |
| ... | 1 | 1 |
| ... | 1 | 1 |
| ... | 1 | 2 |
| ... | 1 | 2 |

SNAP_GET_CONTAINER table function

The SNAP_GET_CONTAINER table function returns the same information as the SNAPCONTAINER administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_TBSP, SNAP_GET_TBSP_PART, SNAP_GET_TBSP_QUIESCER and SNAP_GET_TBSP_RANGE table functions, the SNAP_GET_CONTAINER table function returns data equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 283 on page 1304 for a complete list of information that can be returned.

Syntax

```

>> SNAP_GET_CONTAINER ( ( dbname )
                        [ , member ] )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all members where the database is active.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_CONTAINER table function takes a snapshot for the currently connected database and database member.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_CONTAINER table function

- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve details for the table space containers on the currently connected database on the currently connected database member.

```
SELECT SNAPSHOT_TIMESTAMP, TBSP_NAME, TBSP_ID, CONTAINER_NAME,
       CONTAINER_ID, CONTAINER_TYPE, ACCESSIBLE
FROM TABLE(SNAP_GET_CONTAINER(' ', -1)) AS T
```

The following is an example of output from this query.

| SNAPSHOT_TIMESTAMP | TBSP_NAME | TBSP_ID | ... |
|----------------------------|-------------------|---------|-----|
| 2005-04-25-14.42.10.899253 | SYSCATSPACE | 0 | ... |
| 2005-04-25-14.42.10.899253 | TEMPSPACE1 | 1 | ... |
| 2005-04-25-14.42.10.899253 | USERSPACE1 | 2 | ... |
| 2005-04-25-14.42.10.899253 | SYSTOOLSPACE | 3 | ... |
| 2005-04-25-14.42.10.899253 | MYTEMP | 4 | ... |
| 2005-04-25-14.42.10.899253 | WHATSNEWTEMPSPACE | 5 | ... |

Output from this query (continued).

| ... | CONTAINER_NAME | CONTAINER_ID | ... |
|-----|---------------------------------------|--------------|-----|
| ... | D:\DB2\NODE0000\SQL00002\SQLT0000.0 | 0 | ... |
| ... | D:\DB2\NODE0000\SQL00002\SQLT0001.0 | 0 | ... |
| ... | D:\DB2\NODE0000\SQL00002\SQLT0002.0 | 0 | ... |
| ... | D:\DB2\NODE0000\SQL00002\SYSTOOLSPACE | 0 | ... |
| ... | D:\DB2\NODE0000\SQL003 | 0 | ... |
| ... | d:\DGTTSWhatsNewContainer | 0 | ... |

Output from this query (continued).

| ... | CONTAINER_TYPE | ACCESSIBLE |
|-----|----------------|------------|
| ... | CONT_PATH | 1 |
| ... | CONT_PATH | 1 |
| ... | CONT_PATH | 1 |
| ... | CONT_PATH | 1 |
| ... | CONT_PATH | 1 |
| ... | CONT_PATH | 1 |

Information returned

NOTE: The BUFFERPOOL database manager monitor switch must be turned on in order for the file system information to be returned.

Table 317. Information returned by the SNAPCONTAINER administrative view and the SNAP_GET_CONTAINER table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container name |
| CONTAINER_ID | BIGINT | container_id - Container identification |
| CONTAINER_TYPE | VARCHAR(16) | container_type - Container type. This is a text identifier based on the defines in sqlutil.h and is one of: <ul style="list-style-type: none"> • DISK_EXTENT_TAG • DISK_PAGE_TAG • FILE_EXTENT_TAG • FILE_PAGE_TAG • PATH |
| TOTAL_PAGES | BIGINT | container_total_pages - Total pages in container |
| USABLE_PAGES | BIGINT | container_usable_pages - Usable pages in container |
| ACCESSIBLE | SMALLINT | container_accessible - Accessibility of container |
| STRIPE_SET | BIGINT | container_stripe_set - Stripe set |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| FS_ID | VARCHAR(22) | fs_id - Unique file system identification number |
| FS_TOTAL_SIZE | BIGINT | fs_total_size - Total size of a file system |
| FS_USED_SIZE | BIGINT | fs_used_size - Amount of space used on a file system |

SNAPDB administrative view and SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group

The SNAPDB administrative view and the SNAP_GET_DB table function return snapshot information from the database (dbase) logical group.

Important: The SNAPDB administrative view and SNAP_GET_DB table function are deprecated and have been replaced by the “MON_GET_DATABASE table function - Get database metrics” on page 619 and “MON_GET_TRANSACTION_LOG table function - Get log information” on page 827.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPDB administrative view” on page 1308
- “SNAP_GET_DB table function” on page 1310

SNAPDB administrative view

This administrative view allows you to retrieve snapshot information from the dbase logical group for the currently connected database.

Used in conjunction with ADMIN_GET_STORAGE_PATHS table function, MON_GET_MEMORY_POOL, MON_GET_TRANSACTION_LOG, and MON_GET_HADR, the SNAPDB administrative view provides information equivalent to the **GET SNAPSHOT FOR DATABASE on database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 285 on page 1312 for a complete list of information that is returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPDB administrative view
- CONTROL privilege on the SNAPDB administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DB table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMOINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Examples

Retrieve the status, platform, location, and connect time for all database members of the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME, DB_STATUS, SERVER_PLATFORM,
       DB_LOCATION, DB_CONN_TIME, DBPARTITIONNUM
FROM SYSIBMADM.SNAPDB ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| DB_NAME | DB_STATUS | SERVER_PLATFORM | DB_LOCATION | ... |
|---------|-----------|-----------------|-------------|-----|
| TEST | ACTIVE | AIX64 | LOCAL | ... |
| TEST | ACTIVE | AIX64 | LOCAL | ... |
| TEST | ACTIVE | AIX64 | LOCAL | ... |

3 record(s) selected.

Output from this query (continued).

| ... | DB_CONN_TIME | DBPARTITIONNUM |
|-----|----------------------------|----------------|
| ... | 2006-01-08-16.48.30.665477 | 0 |
| ... | 2006-01-08-16.48.34.005328 | 1 |
| ... | 2006-01-08-16.48.34.007937 | 2 |

This routine can be used by calling the following on the command line:

```
SELECT TOTAL_OLAP_FUNCS, OLAP_FUNC_OVERFLOWS, ACTIVE_OLAP_FUNCS
FROM SYSIBMADM.SNAPDB
```

| TOTAL_OLAP_FUNCS | OLAP_FUNC_OVERFLOWS | ACTIVE_OLAP_FUNCS |
|------------------|---------------------|-------------------|
| 7 | 2 | 1 |

1 record(s) selected.

After running a workload, a user can use the following query:

```
SELECT STATS_CACHE_SIZE, STATS_FABRICATIONS, SYNC_RUNSTATS,
       ASYNC_RUNSTATS, STATS_FABRICATE_TIME, SYNC_RUNSTATS_TIME
FROM SYSIBMADM.SNAPDB
```

| STATS_CACHE_SIZE | STATS_FABRICATIONS | SYNC_RUNSTATS | ASYNC_RUNSTATS | ... |
|------------------|--------------------|---------------|----------------|-----|
| 128 | 2 | 1 | 0 | ... |

| ... | STATS_FABRICATE_TIME | SYNC_RUNSTATS_TIME |
|-----|----------------------|--------------------|
| ... | 10 | 100 |

1 record(s) selected.

SNAP_GET_DB table function

The SNAP_GET_DB table function returns the same information as the SNAPDB administrative view.

Used in conjunction with the ADMIN_GET_STORAGE_PATHS table function, MON_GET_MEMORY_POOL, MON_GET_TRANSACTION_LOG, and MON_GET_HADR table functions, the SNAP_GET_DB table function provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES** CLP command.

Refer to Table 285 on page 1312 for a complete list of information that is returned.

Syntax

→ SNAP_GET_DB ((*dbname* [, *member*]) →

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DB table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DB table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Retrieve the status, platform, location, and connect time as an aggregate view across all database members of the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME, DB_STATUS, SERVER_PLATFORM,
       DB_LOCATION, DB_CONN_TIME FROM TABLE(SNAP_GET_DB('', -2)) AS T
```

The following is an example of output from this query.

| DB_NAME | DB_STATUS | SERVER_PLATFORM | ... |
|---------|-----------|-----------------|-----|
| ----- | ----- | ----- | ... |
| SAMPLE | ACTIVE | AIX64 | ... |

1 record(s) selected.

Output from this query (continued).

| ... | DB_LOCATION | DB_CONN_TIME |
|-----|-------------|----------------------------|
| ... | ----- | ----- |
| ... | LOCAL | 2005-07-24-22.09.22.013196 |

Example 2: Retrieve the status, platform, location, and connect time as an aggregate view across all database members for all active databases in the same instance that contains the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME, DB_STATUS, SERVER_PLATFORM,
       DB_LOCATION, DB_CONN_TIME
FROM TABLE(SNAP_GET_DB(CAST (NULL AS VARCHAR(128)), -2)) AS T
```

The following is an example of output from this query.

| DB_NAME | DB_STATUS | SERVER_PLATFORM | ... |
|---------|-----------|-----------------|-----|
| ----- | ----- | ----- | ... |
| TOOLSDB | ACTIVE | AIX64 | ... |
| SAMPLE | ACTIVE | AIX64 | ... |

Output from this query (continued).

| ... | DB_LOCATION | DB_CONN_TIME |
|-----|-------------|----------------------------|
| ... | ----- | ----- |
| ... | LOCAL | 2005-07-24-22.26.54.396335 |
| ... | LOCAL | 2005-07-24-22.09.22.013196 |

Example 3: This routine can be used by calling the following on the command line:

When connected to a database:

```
SELECT TOTAL_OLAP_FUNCS, OLAP_FUNC_OVERFLOWS, ACTIVE_OLAP_FUNCS
FROM TABLE (SNAP_GET_DB('', 0)) AS T
```

The output will look like:

| TOTAL_OLAP_FUNCS | OLAP_FUNC_OVERFLOWS | ACTIVE_OLAP_FUNCS |
|------------------|---------------------|-------------------|
| ----- | ----- | ----- |
| 7 | 2 | 1 |

1 record(s) selected.

Example 4: After running a workload, a user can use the following query with the table function.

```
SELECT STATS_CACHE_SIZE, STATS_FABRICATIONS, SYNC_RUNSTATS,
       ASYNC_RUNSTATS, STATS_FABRICATE_TIME, SYNC_RUNSTATS_TIME
FROM TABLE (SNAP_GET_DB('mytestdb', -1)) AS SNAPDB
```

STATS_CACHE_SIZE STATS_FABRICATIONS SYNC_RUNSTATS ASYNC_RUNSTATS ...

```
-----
                200                1                2                0 ...
```

Continued

```
...STATS_FABRICATE_TIME  SYNC_RUNSTATS_TIME
-----
...                      2                      32
```

1 record(s) selected.

Example 5: The following example shows how you can use the SNAP_GET_DB table function to determine the status of a database:

```
SELECT SUBSTR
      (DB_NAME, 1, 20) AS DB_NAME, DB_STATUS
FROM   table(SNAP_GET_DB('hadrdb', 0))
```

```
DB_NAME          DB_STATUS
-----
HADRDB           ACTIVE_STANDBY
```

Information returned

Table 318. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function

| Column name | Data type | Description or corresponding monitor element |
|------------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| DB_PATH | VARCHAR(1024) | db_path - Database path |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input database alias |
| DB_STATUS | VARCHAR(16) | db_status - Status of database. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • ACTIVE • QUIESCE_PEND • QUIESCED • ROLLFWD • ACTIVE_STANDBY - the HADR database is in a standby mode with reads on standby enabled. • STANDBY - the HADR database is in standby mode (reads on standby are not enabled). |
| CATALOG_PARTITION | SMALLINT | catalog_node - Catalog database partition number |
| CATALOG_PARTITION_NAME | VARCHAR(128) | catalog_node_name - Catalog database partition network name |

Table 318. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------|-------------|---|
| SERVER_PLATFORM | VARCHAR(12) | <p>server_platform - Server operating system. This interface returns a text identifier based on defines in sqlmon.h, and is one of:</p> <ul style="list-style-type: none"> • AIX • AIX64 • AS400_DRDA • DOS • DYNIX • HP • HP64 • HPIA • HPIA64 • LINUX • LINUX390 • LINUXIA64 • LINUXPPC • LINUXPPC64 • LINUXX8664 • LINUXZ64 • MAC • MVS_DRDA • NT • NT64 • OS2 • OS390 • SCO • SGI • SNI • SUN • SUN64 • UNKNOWN • UNKNOWN_DRDA • VM_DRDA • VSE_DRDA • WINDOWS |
| DB_LOCATION | VARCHAR(12) | <p>db_location - Database location. This interface returns a text identifier based on defines in sqlmon.h, and is one of:</p> <ul style="list-style-type: none"> • LOCAL • REMOTE |
| DB_CONN_TIME | TIMESTAMP | db_conn_time - Database activation timestamp |
| LAST_RESET | TIMESTAMP | last_reset - Last reset timestamp |
| LAST_BACKUP | TIMESTAMP | last_backup - Last backup timestamp |

Table 318. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| CONNECTIONS_TOP | BIGINT | connections_top - Maximum number of concurrent connections |
| TOTAL_CONS | BIGINT | total_cons - Connects since database activation |
| TOTAL_SEC_CONS | BIGINT | total_sec_cons - Secondary connections |
| APPLS_CUR_CONS | BIGINT | appls_cur_cons - Applications connected currently |
| APPLS_IN_DB2 | BIGINT | appls_in_db2 - Applications executing in the database currently |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of associated agents |
| AGENTS_TOP | BIGINT | agents_top - Number of agents created |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum number of coordinating agents |
| LOCKS_HELD | BIGINT | locks_held - Locks held |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_LIST_IN_USE | BIGINT | lock_list_in_use - Total lock list memory in use |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive lock escalations |
| LOCKS_WAITING | BIGINT | locks_waiting - Current agents waiting on locks |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |
| NUM_INDOUBT_TRANS | BIGINT | num_indoubt_trans - Number of indoubt transactions |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total sort heap allocated |
| SORT_SHRHEAP_ALLOCATED | BIGINT | sort_shrheap_allocated - Sort share heap currently allocated |
| SORT_SHRHEAP_TOP | BIGINT | sort_shrheap_top - Sort share heap high water mark |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post shared threshold sorts |
| TOTAL_SORTS | BIGINT | total_sorts - Total sorts |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total sort time |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| ACTIVE_SORTS | BIGINT | active_sorts - Active sorts |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |

Table 318. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer pool asynchronous read time |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time |

Table 318. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|---|
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests |
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer pool no victim buffers |
| POOL_LSN_GAP_CLNS | BIGINT | pool_lsn_gap_clns - Buffer pool log space cleaners triggered |
| POOL_DRTY_PG_STEAL_CLNS | BIGINT | pool_drty_pg_steal_clns - Buffer pool victim page cleaners triggered |
| POOL_DRTY_PG_THRSH_CLNS | BIGINT | pool_drty_pg_thrsh_clns - Buffer pool threshold cleaners triggered |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| FILES_CLOSED | BIGINT | files_closed - Database files closed |
| ELAPSED_EXEC_TIME_S | BIGINT | elapsed_exec_time - Statement execution elapsed time |
| ELAPSED_EXEC_TIME_MS | BIGINT | elapsed_exec_time - Statement execution elapsed time |
| COMMIT_SQL_STMTS | BIGINT | commit_sql_stmts - Commit statements attempted |
| ROLLBACK_SQL_STMTS | BIGINT | rollback_sql_stmts - Rollback statements attempted |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL statements attempted |
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL statements attempted |
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed statement operations |
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL statements executed |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - UPDATE/INSERT/DELETE SQL statements executed |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data definition language (DDL) SQL statements |
| INT_AUTO_REBINDS | BIGINT | int_auto_rebinds - Internal automatic rebinds |

Table 318. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|-----------|--|
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal rows deleted |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal rows inserted |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal rows updated |
| INT_COMMITS | BIGINT | int_commits - Internal commits |
| INT_ROLLBACKS | BIGINT | int_rollback - Internal rollbacks |
| INT_DEADLOCK_ROLLBACKS | BIGINT | int_deadlock_rollback - Internal rollbacks due to deadlock |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated |
| ROWS_SELECTED | BIGINT | rows_selected - Rows selected |
| ROWS_READ | BIGINT | rows_read - Rows read |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/precompiles attempted |
| TOTAL_LOG_AVAILABLE | BIGINT | total_log_available - Total log available |
| TOTAL_LOG_USED | BIGINT | total_log_used - Total log space used |
| SEC_LOG_USED_TOP | BIGINT | sec_log_used_top - Maximum secondary log space used |
| TOT_LOG_USED_TOP | BIGINT | tot_log_used_top - Maximum total log space used |
| SEC_LOGS_ALLOCATED | BIGINT | sec_logs_allocated - Secondary logs allocated currently |
| LOG_READS | BIGINT | log_reads - Number of log pages read |
| LOG_READ_TIME_S | BIGINT | log_read_time - Log read time |
| LOG_READ_TIME_NS | BIGINT | log_read_time - Log read time |
| LOG_WRITES | BIGINT | log_writes - Number of log pages written |
| LOG_WRITE_TIME_S | BIGINT | log_write_time - Log write time |
| LOG_WRITE_TIME_NS | BIGINT | log_write_time - Log write time |
| NUM_LOG_WRITE_IO | BIGINT | num_log_write_io - Number of log writes |
| NUM_LOG_READ_IO | BIGINT | num_log_read_io - Number of log reads |
| NUM_LOG_PART_PAGE_IO | BIGINT | num_log_part_page_io - Number of partial log page writes |
| NUM_LOG_BUFFER_FULL | BIGINT | num_log_buffer_full - Number of full log buffers |
| NUM_LOG_DATA_FOUND_IN_BUFFER | BIGINT | num_log_data_found_in_buffer - Number of log data found in buffer |
| APPL_ID_OLDEST_XACT | BIGINT | appl_id_oldest_xact - Application with oldest transaction |
| LOG_TO_REDO_FOR_RECOVERY | BIGINT | log_to_redo_for_recovery - Amount of log to be redone for recovery |
| LOG_HELD_BY_DIRTY_PAGES | BIGINT | log_held_by_dirty_pages - Amount of log space accounted for by dirty pages |

Table 318. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------------|-----------|--|
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts |
| PKG_CACHE_NUM_OVERFLOWS | BIGINT | pkg_cache_num_overflows - Package cache overflows |
| PKG_CACHE_SIZE_TOP | BIGINT | pkg_cache_size_top - Package cache high water mark |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - Section lookups |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Section inserts |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog cache overflows |
| CAT_CACHE_SIZE_TOP | BIGINT | cat_cache_size_top - Catalog cache high water mark |
| PRIV_WORKSPACE_SIZE_TOP | BIGINT | priv_workspace_size_top - Maximum private workspace size |
| PRIV_WORKSPACE_NUM_OVERFLOWS | BIGINT | priv_workspace_num_overflows - Private workspace overflows |
| PRIV_WORKSPACE_SECTION_INSERTS | BIGINT | priv_workspace_section_inserts - Private workspace section inserts |
| PRIV_WORKSPACE_SECTION_LOOKUPS | BIGINT | priv_workspace_section_lookups - Private workspace section lookups |
| SHR_WORKSPACE_SIZE_TOP | BIGINT | shr_workspace_size_top - Maximum shared workspace size |
| SHR_WORKSPACE_NUM_OVERFLOWS | BIGINT | shr_workspace_num_overflows - Shared workspace overflows |
| SHR_WORKSPACE_SECTION_INSERTS | BIGINT | shr_workspace_section_inserts - Shared workspace section inserts |
| SHR_WORKSPACE_SECTION_LOOKUPS | BIGINT | shr_workspace_section_lookups - Shared workspace section lookups |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total hash joins |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total hash loops |
| HASH_JOIN_OVERFLOWS | BIGINT | hash_join_overflows - Hash join overflows |
| HASH_JOIN_SMALL_OVERFLOWS | BIGINT | hash_join_small_overflows - Hash join small overflows |
| POST_SHRTHRESHOLD_HASH_JOINS | BIGINT | post_shrthreshold_hash_joins - Post threshold hash joins |
| ACTIVE_HASH_JOINS | BIGINT | active_hash_joins - Active hash joins |
| NUM_DB_STORAGE_PATHS | BIGINT | num_db_storage_paths - Number of automatic storage paths |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| SMALLEST_LOG_AVAIL_NODE | INTEGER | smallest_log_avail_node - Node with least available log space |

Table 318. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|-----------|--|
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP functions |
| OLAP_FUNC_OVERFLOWS | BIGINT | olap_func_overflows - OLAP function overflows |
| ACTIVE_OLAP_FUNCS | BIGINT | active_olap_funcs - Active OLAP functions |
| STATS_CACHE_SIZE | BIGINT | stats_cache_size - Size of statistics cache |
| STATS_FABRICATIONS | BIGINT | stats_fabrications - Total number of statistics fabrications |
| SYNC_RUNSTATS | BIGINT | sync_runstats - Total number of synchronous RUNSTATS activities |
| ASYNC_RUNSTATS | BIGINT | async_runstats - Total number of asynchronous RUNSTATS requests |
| STATS_FABRICATE_TIME | BIGINT | stats_fabricate_time - Total time spent on statistics fabrication activities |
| SYNC_RUNSTATS_TIME | BIGINT | sync_runstats_time - Total time spent on synchronous RUNSTATS activities |
| NUM_THRESHOLD_VIOLATIONS | BIGINT | num_threshold_violations - Number of threshold violations |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPDB_MEMORY_POOL administrative view and SNAP_GET_DB_MEMORY_POOL table function - Retrieve database level memory usage information

The SNAPDB_MEMORY_POOL administrative view and the SNAP_GET_DB_MEMORY_POOL table function return information about memory usage at the database level for UNIX platforms only.

The SNAPDB_MEMORY_POOL administrative view and the SNAP_GET_DB_MEMORY_POOL table function return information about memory usage at the database level for UNIX platforms only.

Important: Starting in Version 9.7 Fix Pack 5, the SNAPDB_MEMORY_POOL administrative view and SNAP_GET_DB_MEMORY_POOL table function have been deprecated and replaced by the “MON_GET_MEMORY_POOL - get memory pool information” on page 679 and “MON_GET_MEMORY_SET - get memory set information” on page 682.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPDB_MEMORY_POOL administrative view” on page 1320
- “SNAP_GET_DB_MEMORY_POOL table function” on page 1322

SNAPDB_MEMORY_POOL administrative view

This administrative view allows you to retrieve database level memory usage information for the currently connected database.

Used with the MON_GET_DATABASE, MON_GET_TRANSACTION_LOG, MON_GET_HADR, and ADMIN_GET_STORAGE_PATHS table functions, the SNAPDB_MEMORY_POOL administrative view provides information equivalent to the **GET SNAPSHOT FOR DATABASE ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 286 on page 1323 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- SELECT privilege on the SNAPDB_MEMORY_POOL administrative view
- CONTROL privilege on the SNAPDB_MEMORY_POOL administrative view
- DATAACCESS authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DB_MEMORY_POOL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of memory pools and their current size for the currently connected database, SAMPLE.

```
SELECT POOL_ID, POOL_CUR_SIZE FROM SYSIBMADM.SNAPDB_MEMORY_POOL
```

The following is an example of output from this query.

| POOL_ID | POOL_CUR_SIZE |
|---------------|---------------|
| UTILITY | 32768 |
| PACKAGE_CACHE | 475136 |
| CAT_CACHE | 65536 |
| BP | 2097152 |
| BP | 1081344 |
| BP | 540672 |
| BP | 278528 |
| BP | 147456 |
| BP | 81920 |
| LOCK_MGR | 294912 |
| DATABASE | 3833856 |
| OTHER | 0 |

12 record(s) selected.

SNAP_GET_DB_MEMORY_POOL table function

The SNAP_GET_DB_MEMORY_POOL table function returns the same information as the SNAPDB_MEMORY_POOL administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DB, SNAP_GET_DETAILLOG, SNAP_GET_HADR and ADMIN_GET_STORAGE_PATHS table functions, the SNAP_GET_DB_MEMORY_POOL table function provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES** CLP command.

Refer to Table 286 on page 1323 for a complete list of information that can be returned.

Syntax

```
►► SNAP_GET_DB_MEMORY_POOL (—dbname — [ , member ] ) —————►►
```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DB_MEMORY_POOL table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DB_MEMORY_POOL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of memory pools and their current size for all databases.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, POOL_ID, POOL_CUR_SIZE
FROM TABLE(SNAPSHOT_GET_DB_MEMORY_POOL
(CAST(NULL AS VARCHAR(128)), -1)) AS T
```

The following is an example of output from this query.

| DB_NAME | POOL_ID | POOL_CUR_SIZE |
|---------|---------------|---------------|
| TESTDB | UTILITY | 65536 |
| TESTDB | PACKAGE_CACHE | 851968 |
| TESTDB | CAT_CACHE | 65536 |
| TESTDB | BP | 35913728 |
| TESTDB | BP | 589824 |
| TESTDB | BP | 327680 |
| TESTDB | BP | 196608 |
| TESTDB | BP | 131072 |
| TESTDB | SHARED_SORT | 65536 |
| TESTDB | LOCK_MGR | 10092544 |
| TESTDB | DATABASE | 4980736 |
| TESTDB | OTHER | 196608 |
| SAMPLE | UTILITY | 65536 |
| SAMPLE | PACKAGE_CACHE | 655360 |
| SAMPLE | CAT_CACHE | 131072 |
| SAMPLE | BP | 4325376 |
| SAMPLE | BP | 589824 |
| SAMPLE | BP | 327680 |
| SAMPLE | BP | 196608 |
| SAMPLE | BP | 131072 |
| SAMPLE | SHARED_SORT | 0 |
| SAMPLE | LOCK_MGR | 655360 |
| SAMPLE | DATABASE | 4653056 |
| SAMPLE | OTHER | 196608 |

24 record(s) selected.

Information returned

Table 319. Information returned by the SNAPDB_MEMORY_POOL administrative view and the SNAP_GET_DB_MEMORY_POOL table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |

Table 319. Information returned by the SNAPDB_MEMORY_POOL administrative view and the SNAP_GET_DB_MEMORY_POOL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------|-------------|---|
| POOL_ID | VARCHAR(14) | pool_id - Memory pool identifier. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • APP_GROUP • APPL_CONTROL • APPLICATION • BP • CAT_CACHE • DATABASE • DFM • FCMBP • IMPORT_POOL • LOCK_MGR • MONITOR • OTHER • PACKAGE_CACHE • QUERY • SHARED_SORT • SORT • STATEMENT • STATISTICS • UTILITY |
| POOL_SECONDARY_ID | VARCHAR(32) | pool_secondary_id - Memory pool secondary identifier |
| POOL_CUR_SIZE | BIGINT | pool_cur_size - Current size of memory pool |
| POOL_WATERMARK | BIGINT | pool_watermark - Memory pool watermark |
| POOL_CONFIG_SIZE | BIGINT | pool_config_size - Configured size of memory pool |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPDBM administrative view and SNAP_GET_DBM table function - Retrieve the dbm logical grouping snapshot information

The SNAPDBM administrative view and the SNAP_GET_DBM table function return the snapshot monitor database manager (dbm) logical grouping information.

Important: The SNAPDBM administrative view and SNAP_GET_DBM table function are deprecated and have been replaced by the “MON_GET_INSTANCE table function - Get instance level information” on page 667.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPDBM administrative view” on page 1334
- “SNAP_GET_DBM table function” on page 1336

SNAPDBM administrative view

Used with the SNAPSWITCHES administrative view and the MON_GET_FCM, MON_GET_MEMORY_SET, MON_GET_MEMORY_POOL table functions, the SNAPDBM administrative view provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

The schema is SYSIBMADM.

Refer to Table 288 on page 1337 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPDBM administrative view
- CONTROL privilege on the SNAPDBM administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DBM table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMOINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve database manager status and connection information for all database members.

```
SELECT DB2_STATUS, DB2START_TIME, LAST_RESET, LOCAL_CONS, REM_CONS_IN,  
       (AGENTS_CREATED_EMPTY_POOL/AGENTS_FROM_POOL) AS AGENT_USAGE,  
       DBPARTITIONNUM FROM SYSIBMADM.SNAPDBM ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| DB2_STATUS | DB2START_TIME | LAST_RESET | ... |
|------------|----------------------------|------------|-----|
| ACTIVE | 2006-01-06-14.59.59.059879 | - | ... |
| ACTIVE | 2006-01-06-14.59.59.097605 | - | ... |
| ACTIVE | 2006-01-06-14.59.59.062798 | - | ... |

3 record(s) selected. ...

Output from this query (continued).

| ... | LOCAL_CONS | REM_CONS_IN | AGENT_USAGE | DBPARTITIONNUM |
|-----|------------|-------------|-------------|----------------|
| ... | 1 | 1 | 0 | 0 |
| ... | 0 | 0 | 0 | 1 |
| ... | 0 | 0 | 0 | 2 |

SNAP_GET_DBM table function

The SNAP_GET_DBM table function returns the same information as the SNAPDBM administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DBM_MEMORY_POOL, SNAP_GET_FCM, SNAP_GET_FCM_PART and SNAP_GET_SWITCHES table functions, the SNAP_GET_DBM table function provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

Refer to Table 288 on page 1337 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_DBM (member) ◀◀

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DBM table function calls the snapshot from memory.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DBM table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve the start time and current status of database member number 2.

```
SELECT DB2START_TIME, DB2_STATUS FROM TABLE(SNAP_GET_DBM(2)) AS T
```

The following is an example of output from this query.

```
DB2START_TIME          DB2_STATUS
-----
2006-01-06-14.59.59.062798 ACTIVE
```

Information returned

Table 320. Information returned by the SNAPDBM administrative view and the SNAP_GET_DBM table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total sort heap allocated |
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post threshold sorts |
| PIPED_SORTS_REQUESTED | BIGINT | pipedsortsrequested - Piped sorts requested |
| PIPED_SORTS_ACCEPTED | BIGINT | pipedsortsaccepted - Piped sorts accepted |
| REM_CONS_IN | BIGINT | rem_cons_in - Remote connections to database manager |
| REM_CONS_IN_EXEC | BIGINT | rem_cons_in_exec - Remote Connections Executing in the Database Manager monitor element |
| LOCAL_CONS | BIGINT | local_cons - Local connections |
| LOCAL_CONS_IN_EXEC | BIGINT | local_cons_in_exec - Local Connections Executing in the Database Manager monitor element |
| CON_LOCAL_DBASES | BIGINT | con_local_dbases - Local databases with current connects |

Table 320. Information returned by the SNAPDBM administrative view and the SNAP_GET_DBM table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|--|
| AGENTS_REGISTERED | BIGINT | agents_registered - Agents registered |
| AGENTS_WAITING_ON_TOKEN | BIGINT | agents_waiting_on_token - Agents waiting for a token |
| DB2_STATUS | VARCHAR(12) | db2_status - Status of the database instance

This interface returns a text identifier based on definitions in sqlmon.h, and is one of: <ul style="list-style-type: none"> • ACTIVE • QUIESCE_PEND • QUIESCED |
| AGENTS_REGISTERED_TOP | BIGINT | agents_registered_top - Maximum number of agents registered |
| AGENTS_WAITING_TOP | BIGINT | agents_waiting_top - Maximum number of agents waiting |
| COMM_PRIVATE_MEM | BIGINT | comm_private_mem - Committed private memory |
| IDLE_AGENTS | BIGINT | idle_agents - Number of idle agents |
| AGENTS_FROM_POOL | BIGINT | agents_from_pool - Agents assigned from pool |
| AGENTS_CREATED_EMPTY_POOL | BIGINT | agents_created_empty_pool - Agents created due to empty agent pool |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum number of coordinating agents |
| MAX_AGENT_OVERFLOW | BIGINT | max_agent_overflows - Maximum agent overflows |
| AGENTS_STOLEN | BIGINT | agents_stolen - Stolen agents |
| GW_TOTAL_CONS | BIGINT | gw_total_cons - Total number of attempted connections for Db2 Connect |
| GW_CUR_CONS | BIGINT | gw_cur_cons - Current number of connections for Db2 Connect |
| GW_CONS_WAIT_HOST | BIGINT | gw_cons_wait_host - Number of connections waiting for the host to reply |
| GW_CONS_WAIT_CLIENT | BIGINT | gw_cons_wait_client - Number of connections waiting for the client to send request |
| POST_THRESHOLD_HASH_JOINS | BIGINT | post_threshold_hash_joins - Hash join threshold |
| NUM_GW_CONN_SWITCHES | BIGINT | num_gw_conn_switches - Connection switches |
| DB2START_TIME | TIMESTAMP | db2start_time - Start database manager timestamp |
| LAST_RESET | TIMESTAMP | last_reset - Last reset timestamp |
| NUM_NODES_IN_DB2_INSTANCE | INTEGER | num_nodes_in_db2_instance - Number of nodes in database partition |
| PRODUCT_NAME | VARCHAR(32) | product_name - Product name |

Table 320. Information returned by the SNAPDBM administrative view and the SNAP_GET_DBM table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|--|
| SERVICE_LEVEL | VARCHAR(18) | service_level - Service level |
| SORT_HEAP_TOP | BIGINT | sort_heap_top - Sort private heap high water mark |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| POST_THRESHOLD_OLAP_FUNCS | BIGINT | post_threshold_olap_funcs - OLAP function threshold |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPDBM_MEMORY_POOL administrative view and SNAP_GET_DBM_MEMORY_POOL table function - Retrieve database manager level memory usage information

The SNAPDBM_MEMORY_POOL administrative view and the SNAP_GET_DBM_MEMORY_POOL table function return information about memory usage at the database manager.

Important: Starting in Version 9.7 Fix Pack 5, the SNAPDBM_MEMORY_POOL administrative view and SNAP_GET_DBM_MEMORY_POOL table function have been deprecated and replaced by the “MON_GET_MEMORY_POOL - get memory pool information” on page 679 and “MON_GET_MEMORY_SET - get memory set information” on page 682.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPDBM_MEMORY_POOL administrative view” on page 1339
- “SNAP_GET_DBM_MEMORY_POOL table function” on page 1340

SNAPDBM_MEMORY_POOL administrative view

Used with the SNAPSWITCHES administrative view and the MON_GET_FCM, MON_GET_INSTANCE, MON_GET_MEMORY_SET, and MON_GET_MEMORY_POOL table functions, the SNAPDBM_MEMORY_POOL administrative view provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

The schema is SYSIBMADM.

Refer to Table 289 on page 1341 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- SELECT privilege on the SNAPDBM_MEMORY_POOL administrative view
- CONTROL privilege on the SNAPDBM_MEMORY_POOL administrative view
- DATAACCESS authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DBM_MEMORY_POOL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of the memory pools and their current size for the database manager of the connected database.

```
SELECT POOL_ID, POOL_CUR_SIZE FROM SNAPDBM_MEMORY_POOL
```

The following is an example of output from this query.

| POOL_ID | POOL_CUR_SIZE |
|---------|---------------|
| MONITOR | 65536 |
| OTHER | 29622272 |
| FCMBP | 57606144 |
| ... | |

SNAP_GET_DBM_MEMORY_POOL table function

The SNAP_GET_DBM_MEMORY_POOL table function returns the same information as the SNAPDBM_MEMORY_POOL administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DBM, SNAP_GET_FCM, SNAP_GET_FCM_PART and SNAP_GET_SWITCHES table functions, the SNAP_GET_DBM_MEMORY_POOL table function provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

Refer to Table 289 on page 1341 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_DBM_MEMORY_POOL (member) ►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DBM_MEMORY_POOL table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DBM_MEMORY_POOL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of the memory pools and their current size for all database partitions of the database manager of the connected database.

```
SELECT POOL_ID, POOL_CUR_SIZE, DBPARTITIONNUM
FROM TABLE(SYSPROC.SNAP_GET_DBM_MEMORY_POOL())
AS T ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| POOL_ID | POOL_CUR_SIZE | DBPARTITIONNUM |
|---------|---------------|----------------|
| MONITOR | 65536 | 0 |
| OTHER | 29622272 | 0 |
| FCMBP | 57606144 | 0 |
| MONITOR | 65536 | 1 |
| OTHER | 29425664 | 1 |
| FCMBP | 57606144 | 1 |
| MONITOR | 65536 | 2 |
| OTHER | 29425664 | 2 |
| FCMBP | 57606144 | 2 |

Information returned

Table 321. Information returned by the `SNAPDBM_MEMORY_POOL` administrative view and the `SNAP_GET_DBM_MEMORY_POOL` table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| POOL_ID | VARCHAR(14) | pool_id - Memory pool identifier. This interface returns a text identifier based on defines in <code>sqlmon.h</code> , and is one of: <ul style="list-style-type: none">• APP_GROUP• APPL_CONTROL• APPLICATION• BP• CAT_CACHE• DATABASE• DFM• FCMBP• IMPORT_POOL• LOCK_MGR• MONITOR• OTHER• PACKAGE_CACHE• QUERY• SHARED_SORT• SORT• STATEMENT• STATISTICS• UTILITY |
| POOL_CUR_SIZE | BIGINT | pool_cur_size - Current size of memory pool |
| POOL_WATERMARK | BIGINT | pool_watermark - Memory pool watermark |
| POOL_CONFIG_SIZE | BIGINT | pool_config_size - Configured size of memory pool |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG table function - Retrieve snapshot information from the detail_log logical data group

The `SNAPDETAILLOG` administrative view and the `SNAP_GET_DETAILLOG` table function return snapshot information from the `detail_log` logical data group.

Important: The SNAPDETAILLOG administrative view and the SNAP_GET_DETAILLOG table function are deprecated and have been replaced by the “MON_GET_TRANSACTION_LOG table function - Get log information” on page 827.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPDETAILLOG administrative view” on page 1346
- “SNAP_GET_DETAILLOG table function” on page 1347

SNAPDETAILLOG administrative view

This administrative view allows you to retrieve snapshot information from the detail_log logical data group for the currently connected database.

Used in conjunction with ADMIN_GET_STORAGE_PATHS, MON_GET_DATABASE, MON_GET_HADR, MON_GET_MEMORY_POOL, and MON_GET_MEMORY_SET table functions, the SNAPDETAILLOG administrative view provides information equivalent to the **GET SNAPSHOT FOR DATABASE on database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 291 on page 1348 for a complete list of information that is returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPDETAILLOG administrative view
- CONTROL privilege on the SNAPDETAILLOG administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DETAILLOG table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve log information for all database members for the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 8) AS DB_NAME, FIRST_ACTIVE_LOG,
      LAST_ACTIVE_LOG, CURRENT_ACTIVE_LOG, CURRENT_ARCHIVE_LOG,
      DBPARTITIONNUM
FROM SYSIBMADM.SNAPDETAILOG ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| DB_NAME | FIRST_ACTIVE_LOG | LAST_ACTIVE_LOG | ... |
|---------|------------------|-----------------|-----|
| TEST | 0 | 8 | ... |
| TEST | 0 | 8 | ... |
| TEST | 0 | 8 | ... |

3 record(s) selected.

Output from this query (continued).

| ... | CURRENT_ACTIVE_LOG | CURRENT_ARCHIVE_LOG | DBPARTITIONNUM |
|-----|--------------------|---------------------|----------------|
| ... | 0 | - | 0 |
| ... | 0 | - | 1 |
| ... | 0 | - | 2 |

SNAP_GET_DETAILOG table function

The SNAP_GET_DETAILOG table function returns the same information as the SNAPDETAILOG administrative view.

Used in conjunction with ADMIN_GET_STORAGE_PATHS, MON_GET_HADR, MON_GET_MEMORY_POOL, and SNAP_GET_DB, the SNAPDETAILOG administrative view provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES** CLP command.

Refer to Table 291 on page 1348 for a complete list of information that is returned.

Syntax

►► SNAP_GET_DETAILOG ((*dbname* , *member*)) ►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an

aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DETAILLOG table function takes a snapshot for the currently connected database and database member number.

Authorization

- One of the following authorizations is required:
- EXECUTE privilege on the SNAP_GET_DETAILLOG table function
 - DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve log information for database member 1 for the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 8) AS DB_NAME, FIRST_ACTIVE_LOG,
       LAST_ACTIVE_LOG, CURRENT_ACTIVE_LOG, CURRENT_ARCHIVE_LOG
FROM TABLE(SNAP_GET_DETAILLOG('I', 1)) AS T
```

The following is an example of output from this query.

| DB_NAME | FIRST_ACTIVE_LOG | LAST_ACTIVE_LOG | ... |
|---------|------------------|-----------------|-----|
| ----- | ----- | ----- | ... |
| TEST | 0 | 8 | ... |

1 record(s) selected.

Output from this query (continued).

| ... | CURRENT_ACTIVE_LOG | CURRENT_ARCHIVE_LOG |
|-----|--------------------|---------------------|
| ... | ----- | ----- |
| ... | 0 | - |

Information returned

Table 322. Information returned by the SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG table function

| Column name | Data type | Description or corresponding monitor element |
|---------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| FIRST_ACTIVE_LOG | BIGINT | first_active_log - First active log file number |
| LAST_ACTIVE_LOG | BIGINT | last_active_log - Last active log file number |
| CURRENT_ACTIVE_LOG | BIGINT | current_active_log - Current active log file number |
| CURRENT_ARCHIVE_LOG | BIGINT | current_archive_log - Current archive log file number |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPFCM administrative view and SNAP_GET_FCM table function - Retrieve the fcm logical data group snapshot information

The SNAPFCM administrative view and the SNAP_GET_FCM table function return information about the fast communication manager from a database manager snapshot, in particular, the fcm logical data group.

Important: The SNAPFCM administrative view and the SNAP_GET_FCM table function are deprecated and have been replaced by the “MON_GET_FCM - Get FCM metrics” on page 652.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPFCM administrative view” on page 1355
- “SNAP_GET_FCM table function” on page 1356

SNAPFCM administrative view

Used with the SNAPSWITCHES administrative view and the MON_GET_FCM, MON_GET_INSTANCE, MON_GET_MEMORY_SET, and MON_GET_MEMORY_POOL table functions, the SNAPFCM administrative view provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

The schema is SYSIBMADM.

Refer to Table 294 on page 1357 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPFCM administrative view
- CONTROL privilege on the SNAPFCM administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_FCM table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve information about the fast communication manager's message buffers on all members.

```
SELECT BUFF_FREE, BUFF_FREE_BOTTOM, MEMBER
FROM SYSIBADM.SNAPFCM ORDER BY MEMBER
```

The following is an example of output from this query.

| BUFF_FREE | BUFF_FREE_BOTTOM | MEMBER |
|-----------|------------------|--------|
| 5120 | 5100 | 0 |
| 5120 | 5100 | 1 |
| 5120 | 5100 | 2 |

SNAP_GET_FCM table function

The SNAP_GET_FCM table function returns the same information as the SNAPFCM administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DBM, SNAP_GET_DBM_MEMORY_POOL, SNAP_GET_FCM_PART and SNAP_GET_SWITCHES table functions, the SNAP_GET_FCM table function provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

Refer to Table 294 on page 1357 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_FCM (member) ►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current member, or -2 for an aggregate of all active members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_FCM table function takes a snapshot for the currently connected database and database member.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_FCM table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve information about the fast communication manager's message buffers on database member 1.

```
SELECT BUFF_FREE, BUFF_FREE_BOTTOM, MEMBER  
FROM TABLE(SYSPROC.SNAP_GET_FCM( 1 )) AS T
```

The following is an example of output from this query.

| | | |
|-----------|------------------|--------|
| BUFF_FREE | BUFF_FREE_BOTTOM | MEMBER |
| 5120 | 5100 | 1 |

Information returned

Table 323. Information returned by the SNAPFCM administrative view and the SNAP_GET_FCM table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| BUFF_FREE | BIGINT | buff_free - FCM buffers currently free |
| BUFF_FREE_BOTTOM | BIGINT | buff_free_bottom - Minimum FCM Buffers Free |
| CH_FREE | BIGINT | ch_free - Channels Currently Free |
| CH_FREE_BOTTOM | BIGINT | ch_free_bottom - Minimum Channels Free |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function - Retrieve the fcm_node logical data group snapshot information

The SNAPFCM_PART administrative view and the SNAP_GET_FCM_PART table function return information about the fast communication manager from a database manager snapshot, in particular, the fcm_node logical data group.

Important: The SNAPFCM_PART administrative view and the SNAP_GET_FCM_PART table function are deprecated and have been replaced by the “MON_GET_FCM - Get FCM metrics” on page 652.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPFCM_PART administrative view” on page 1358
- “SNAP_GET_FCM_PART table function” on page 1359

SNAPFCM_PART administrative view

Used with the SNAPSWITCHES administrative view and the MON_GET_FCM, MON_GET_INSTANCE, MON_GET_MEMORY_SET, and MON_GET_MEMORY_POOL table functions, the SNAPFCM_PART administrative view provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

The schema is SYSIBMADM.

Refer to Table 295 on page 1360 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPFCM_PART administrative view
- CONTROL privilege on the SNAPFCM_PART administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_FCM_PART table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve buffers sent and received information for the fast communication manager.

```
SELECT CONNECTION_STATUS, TOTAL_BUFFERS_SENT, TOTAL_BUFFERS_RECEIVED
FROM SYSIBMADM.SNAPFCM_PART WHERE MEMBER = 0
```

The following is an example of output from this query.

| CONNECTION_STATUS | TOTAL_BUFFERS_SENT | TOTAL_BUFFERS_RCVD |
|-------------------|--------------------|--------------------|
| INACTIVE | 2 | 1 |

1 record(s) selected.

SNAP_GET_FCM_PART table function

The SNAP_GET_FCM_PART table function returns the same information as the SNAPFCM_PART administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DBM, SNAP_GET_DBM_MEMORY_POOL, SNAP_GET_FCM and SNAP_GET_SWITCHES table functions, the SNAP_GET_FCM_PART table function provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

Refer to Table 295 on page 1360 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_FCM_PART (*member*) ►►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current member. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_FCM_PART table function takes a snapshot for the currently connected database and member.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_FCM_PART table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve buffers sent and received information for the fast communication manager for all members.

```
SELECT FCM_MEMBER, TOTAL_BUFFERS_SENT, TOTAL_BUFFERS_RCVD,  
       MEMBER FROM TABLE(SNAP_GET_FCM_PART()) AS T  
ORDER BY MEMBER
```

The following is an example of output from this query.

| FCM_MEMBER | TOTAL_BUFFERS_SENT | TOTAL_BUFFERS_RCVD | MEMBER |
|------------|--------------------|--------------------|--------|
| 0 | 305 | 305 | 0 |
| 1 | 5647 | 1664 | 0 |
| 2 | 5661 | 1688 | 0 |
| 0 | 19 | 19 | 1 |
| 1 | 305 | 301 | 1 |
| 2 | 1688 | 5661 | 1 |
| 0 | 1664 | 5647 | 2 |
| 1 | 10 | 10 | 2 |
| 2 | 301 | 305 | 2 |

Information returned

Table 324. Information returned by the SNAPFCM_PART administrative view and the SNAP_GET_FCM_PART table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| CONNECTION_STATUS | VARCHAR(10) | connection_status - Connection status. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • INACTIVE • ACTIVE • CONGESTED |
| TOTAL_BUFFERS_SENT | BIGINT | total_buffers_sent - Total FCM buffers sent |
| TOTAL_BUFFERS_RCVD | BIGINT | total_buffers_rcvd - Total FCM buffers received |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| FCM_DBPARTITIONNUM | SMALLINT | The database partition number to which data was sent or from which data was received (as per the TOTAL_BUFFERS_SENT and TOTAL_BUFFERS_RCVD columns). |
| MEMBER | SMALLINT | member - Database member monitor element |
| FCM_MEMBER | SMALLINT | The member to which data was sent or from which data was received (as per the TOTAL_BUFFERS_SENT and TOTAL_BUFFERS_RCVD columns). |

SNAPHADR administrative view and SNAP_GET_HADR table function - Retrieve hadr logical data group snapshot information

The SNAPHADR administrative view and the SNAP_GET_HADR table function return information about high availability disaster recovery from a database snapshot, in particular, the hadr logical data group.

The SNAPHADR administrative view and SNAP_GET_HADR table function have been deprecated and replaced by the “MON_GET_HADR table function - Returns high availability disaster recovery (HADR) monitoring information” on page 656.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPHADR administrative view” on page 1361
- “SNAP_GET_HADR table function” on page 1362

SNAPHADR administrative view

This administrative view allows you to retrieve hadr logical data group snapshot information for the currently connected database. The data is only returned by this view if the database is a primary or standby high availability disaster recovery (HADR) database.

Used with the MON_GET_DATABASE, MON_GET_MEMORY_SET, MON_GET_MEMORY_POOL, MON_GET_TRANSACTION_LOG and ADMIN_GET_STORAGE_PATHS table functions, the SNAPHADR administrative view provides information equivalent to the **GET SNAPSHOT FOR DATABASE ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 296 on page 1364 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- SELECT privilege on the SNAPHADR administrative view
- CONTROL privilege on the SNAPHADR administrative view
- DATAACCESS authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_HADR table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve the configuration and status information for HADR on the primary HADR database.

```
SELECT SUBSTR(DB_NAME, 1, 8) AS DBNAME, HADR_ROLE, HADR_STATE,
       HADR_SYNCMODE, HADR_CONNECT_STATUS
FROM SYSIBMADM.SNAPHADR
```

The following is an example of output from this query.

| DBNAME | HADR_ROLE | HADR_STATE | HADR_SYNCMODE | HADR_CONNECT_STATUS |
|--------|-----------|------------|---------------|---------------------|
| SAMPLE | PRIMARY | PEER | SYNC | CONNECTED |

1 record(s) selected.

SNAP_GET_HADR table function

The SNAP_GET_HADR table function returns the same information as the SNAPHADR administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DB, SNAP_GET_DB_MEMORY_POOL, SNAP_GET_DETAILLOG and ADMIN_GET_STORAGE_PATHS table functions, the SNAP_GET_HADR table function provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES** CLP command.

Refer to Table 296 on page 1364 for a complete list of information that can be returned.

Syntax

►► SNAP_GET_HADR ((*dbname* [, *member*])) ►►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_HADR table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_HADR table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve the configuration and status information for HADR for all databases.

```
SELECT SUBSTR(DB_NAME, 1, 8) AS DBNAME, HADR_ROLE, HADR_STATE,
       HADR_SYNCMODE, HADR_CONNECT_STATUS
FROM TABLE (SNAP_GET_HADR (CAST (NULL as VARCHAR(128)), 0)) as T
```

The following is an example of output from this query.

| DBNAME | HADR_ROLE | HADR_STATE | HADR_SYNCMODE | HADR_CONNECT_STATUS |
|--------|-----------|--------------|---------------|---------------------|
| SAMPLE | PRIMARY | PEER | SYNC | CONNECTED |
| TESTDB | PRIMARY | DISCONNECTED | NEARSYNC | DISCONNECTED |

2 record(s) selected.

Information returned

Table 325. Information returned by the SNAPHADR administrative view and the SNAP_GET_HADR table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |

Table 325. Information returned by the SNAPHADR administrative view and the SNAP_GET_HADR table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|--------------|--|
| HADR_ROLE | VARCHAR(10) | hadr_role - HADR role. This interface returns a text identifier based on the defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • PRIMARY • STANDARD • STANDBY |
| HADR_STATE | VARCHAR(14) | hadr_state - HADR state. This interface returns a text identifier based on the defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • DISCONNECTED • LOCAL_CATCHUP • PEER • REM_CATCH_PEN • REM_CATCHUP |
| HADR_SYNCMODE | VARCHAR(10) | hadr_syncmode - HADR synchronization mode. This interface returns a text identifier based on the defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • ASYNC • NEARSYNC • SUPERASYNC • SYNC |
| HADR_CONNECT_STATUS | VARCHAR(12) | hadr_connect_status - HADR connection status. This interface returns a text identifier based on the defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • CONGESTED • CONNECTED • DISCONNECTED |
| HADR_CONNECT_TIME | TIMESTAMP | hadr_connect_time - HADR connection time |
| HADR_HEARTBEAT | INTEGER | hadr_heartbeat - HADR heartbeat |
| HADR_LOCAL_HOST | VARCHAR(255) | hadr_local_host - HADR local host |
| HADR_LOCAL_SERVICE | VARCHAR(40) | hadr_local_service - HADR local service |
| HADR_REMOTE_HOST | VARCHAR(255) | hadr_remote_host - HADR remote host |
| HADR_REMOTE_SERVICE | VARCHAR(40) | hadr_remote_service - HADR remote service |
| HADR_REMOTE_INSTANCE | VARCHAR(128) | hadr_remote_instance - HADR remote instance |
| HADR_TIMEOUT | BIGINT | hadr_timeout - HADR timeout |

Table 325. Information returned by the SNAPHADR administrative view and the SNAP_GET_HADR table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|--------------|--|
| HADR_PRIMARY_LOG_FILE | VARCHAR(255) | hadr_primary_log_file - HADR primary log file |
| HADR_PRIMARY_LOG_PAGE | BIGINT | hadr_primary_log_page - HADR primary log page |
| HADR_PRIMARY_LOG_LSN | BIGINT | hadr_primary_log_lsn - HADR primary log LSN |
| HADR_STANDBY_LOG_FILE | VARCHAR(255) | hadr_standby_log_file - HADR standby log file |
| HADR_STANDBY_LOG_PAGE | BIGINT | hadr_standby_log_page - HADR standby log page |
| HADR_STANDBY_LOG_LSN | BIGINT | hadr_standby_log_lsn - HADR standby log LSN |
| HADR_LOG_GAP | BIGINT | hadr_log_gap - HADR log gap |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPLOCK administrative view and SNAP_GET_LOCK table function - Retrieve lock logical data group snapshot information

The SNAPLOCK administrative view and the SNAP_GET_LOCK table function return snapshot information about locks, in particular, the lock logical data group.

Note: This administrative view and table function have been deprecated and replaced by the “MON_GET_APPL_LOCKWAIT - Get information about locks for which an application is waiting” on page 561, “MON_GET_LOCKS - List all locks in the currently connected database” on page 674, and “MON_FORMAT_LOCK_NAME - Format the internal lock name and return details” on page 483.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPLOCK administrative view” on page 1366
- “SNAP_GET_LOCK table function” on page 1367

SNAPLOCK administrative view

This administrative view allows you to retrieve lock logical data group snapshot information for the currently connected database.

Used with the SNAPLOCKWAIT administrative view, the SNAPLOCK administrative view provides information equivalent to the **GET SNAPSHOT FOR LOCKS ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 297 on page 1368 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- SELECT privilege on the SNAPLOCK administrative view
- CONTROL privilege on the SNAPLOCK administrative view
- DATAACCESS authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_LOCK table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve lock information for the database member 0 of the currently connected database.

```
SELECT AGENT_ID, LOCK_OBJECT_TYPE, LOCK_MODE, LOCK_STATUS
FROM SYSIBMADM.SNAPLOCK WHERE DBPARTITIONNUM = 0
```

The following is an example of output from this query.

| AGENT_ID | LOCK_OBJECT_TYPE | LOCK_MODE | LOCK_STATUS |
|----------|------------------|-----------|-------------|
| 7 | TABLE | IX | GRNT |

1 record(s) selected.

SNAP_GET_LOCK table function

The SNAP_GET_LOCK table function returns the same information as the SNAPLOCK administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_LOCKWAIT table function, the SNAP_GET_LOCK table function provides information equivalent to the **GET SNAPSHOT FOR LOCKS ON database-alias** CLP command.

Refer to Table 297 on page 1368 for a complete list of information that can be returned.

Syntax

▶▶ SNAP_GET_LOCK ((*dbname* [, *member*])) ▶▶

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify a null value or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_LOCK table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_LOCK table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve lock information for the current database member of the currently connected database.

```
SELECT AGENT_ID, LOCK_OBJECT_TYPE, LOCK_MODE, LOCK_STATUS
FROM TABLE(SNAP_GET_LOCK('T',-1)) as T
```

The following is an example of output from this query.

| AGENT_ID | LOCK_OBJECT_TYPE | LOCK_MODE | LOCK_STATUS |
|----------|------------------|-----------|-------------|
| 680 | INTERNALV_LOCK | S | GRNT |
| 680 | INTERNALP_LOCK | S | GRNT |

2 record(s) selected.

Information returned

Table 326. Information returned by the SNAPLOCK administrative view and the SNAP_GET_LOCK table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| TAB_FILE_ID | BIGINT | table_file_id - Table file identification |
| LOCK_OBJECT_TYPE | VARCHAR(18) | lock_object_type - Lock object type waited on. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none">AUTORESIZE_LOCKAUTOSTORAGE_LOCKBLOCK_LOCKEOT_LOCKINPLACE_REORG_LOCKINTERNAL_LOCKINTERNALB_LOCKINTERNALC_LOCKINTERNALJ_LOCKINTERNALL_LOCKINTERNALO_LOCKINTERNALQ_LOCKINTERNALP_LOCKINTERNALS_LOCKINTERNALT_LOCKINTERNALV_LOCKKEYVALUE_LOCKROW_LOCKSYSBOOT_LOCKTABLE_LOCKTABLE_PART_LOCKTABLESPACE_LOCKXML_PATH_LOCKCOL_TABLE_SERIALIZE_LOCK |

Table 326. Information returned by the SNAPLOCK administrative view and the SNAP_GET_LOCK table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------|--------------|---|
| LOCK_MODE | VARCHAR(10) | lock_mode - Lock mode. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |
| LOCK_STATUS | VARCHAR(10) | lock_status - Lock status. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • CONV • GRNT |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock escalation |
| TABNAME | VARCHAR(128) | table_name - Table name |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| LOCK_ATTRIBUTES | VARCHAR(128) | lock_attributes - Lock attributes. This interface returns a text identifier based on the defines in sqlmon.h. If there are no locks, the text identifier is NONE, otherwise, it is any combination of the following separated by a '+' sign: <ul style="list-style-type: none"> • ALLOW_NEW • DELETE_IN_BLOCK • ESCALATED • INSERT • NEW_REQUEST • RR • RR_IN_BLOCK • UPDATE_DELETE • WAIT_FOR_AVAIL |
| LOCK_COUNT | BIGINT | lock_count - Lock count |

Table 326. Information returned by the SNAPLOCK administrative view and the SNAP_GET_LOCK table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|--|
| LOCK_CURRENT_MODE | VARCHAR(10) | lock_current_mode - Original lock mode before conversion. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |
| LOCK_HOLD_COUNT | BIGINT | lock_hold_count - Lock hold count |
| LOCK_NAME | VARCHAR(32) | lock_name - Lock name |
| LOCK_RELEASE_FLAGS | BIGINT | lock_release_flags - Lock release flags |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition identifier. For a non-partitioned table, this element is NULL. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function - Retrieve lockwait logical data group snapshot information

The SNAPLOCKWAIT administrative view and the SNAP_GET_LOCKWAIT table function return snapshot information about lock waits, in particular, the lockwait logical data group.

Note: This administrative view and table function have been deprecated and replaced by the “MON_LOCKWAITS administrative view - Retrieve metrics for applications that are waiting to obtain locks” on page 441 and the “MON_GET_APPL_LOCKWAIT - Get information about locks for which an application is waiting” on page 561, “MON_GET_LOCKS - List all locks in the currently connected database” on page 674, and “MON_FORMAT_LOCK_NAME - Format the internal lock name and return details” on page 483.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPLOCKWAIT administrative view” on page 1371

- “SNAP_GET_LOCKWAIT table function” on page 1372

SNAPLOCKWAIT administrative view

This administrative view allows you to retrieve lockwait logical data group snapshot information for the currently connected database.

Used with the SNAPLOCK administrative view, the SNAPLOCKWAIT administrative view provides information equivalent to the **GET SNAPSHOT FOR LOCKS ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 298 on page 1374 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required:

- SELECT privilege on the SNAPLOCKWAIT administrative view
- CONTROL privilege on the SNAPLOCKWAIT administrative view
- DATAACCESS authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_LOCKWAIT table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve lock wait information for database member 0 for the currently connected database.

```
SELECT AGENT_ID, LOCK_MODE, LOCK_OBJECT_TYPE, AGENT_ID_HOLDING_LK,
       LOCK_MODE_REQUESTED FROM SYSIBMADM.SNAPLOCKWAIT
WHERE DBPARTITIONNUM = 0
```

The following is an example of output from this query.

| AGENT_ID | LOCK_MODE | LOCK_OBJECT_TYPE | ... |
|----------|-----------|------------------|-----|
| ----- | ----- | ----- | ... |
| 7 | IX | TABLE | ... |

1 record(s) selected.

Output from this query (continued).

```
... AGENT_ID_HOLDING_LK  LOCK_MODE_REQUESTED
... -----
...                      12 IS
```

SNAP_GET_LOCKWAIT table function

The SNAP_GET_LOCKWAIT table function returns the same information as the SNAPLOCKWAIT administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_LOCK table function, the SNAP_GET_LOCKWAIT table function provides information equivalent to the **GET SNAPSHOT FOR LOCKS ON database-alias** CLP command.

Refer to Table 298 on page 1374 for a complete list of information that can be returned.

Syntax

```
➤ SNAP_GET_LOCKWAIT (—dbname—, —member—)
```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify a null value or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_LOCKWAIT table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_LOCKWAIT table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve lock wait information for current database member for the currently connected database.

```
SELECT AGENT_ID, LOCK_MODE, LOCK_OBJECT_TYPE, AGENT_ID_HOLDING_LK,
       LOCK_MODE_REQUESTED FROM TABLE(SNAP_GET_LOCKWAIT('',-1)) AS T
```

The following is an example of output from this query.

```
AGENT_ID      LOCK_MODE  LOCK_OBJECT_TYPE  ...
-----
12 X          ROW_LOCK      ...
```

1 record(s) selected.

Output from this query (continued).

```
... AGENT_ID_HOLDING_LK  LOCK_MODE_REQUESTED
... -----
...                      7 X
```

Usage note

To see lock wait information, you must first turn on the default LOCK monitor switch in the database manager configuration. To have the change take effect immediately explicitly attach to the instance using CLP and then issue the CLP command:

```
UPDATE DATABASE MANAGER CONFIGURATION CLP USING DFT_MON_LOCK ON
```

The default setting can also be turned on through the ADMIN_CMD stored procedure. For example:

```
CALL SYSPROC.ADMIN_CMD('update dbm cfg using DFT_MON_LOCK ON')
```

If the ADMIN_CMD stored procedure is used or if the clp command is used without having previously attached to the instance, the instance must be recycled before the change takes effect.

Information returned

Table 327. Information returned by the SNAPLOCKWAIT administrative view and the SNAP_GET_LOCKWAIT table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |

Table 327. Information returned by the `SNAPLOCKWAIT` administrative view and the `SNAP_GET_LOCKWAIT` table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------|-------------|---|
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| SUBSECTION_NUMBER | BIGINT | ss_number - Subsection number |
| LOCK_MODE | VARCHAR(10) | lock_mode - Lock mode. This interface returns a text identifier based on the defines in <code>sqlmon.h</code> and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |
| LOCK_OBJECT_TYPE | VARCHAR(18) | lock_object_type - Lock object type waited on. This interface returns a text identifier based on the defines in <code>sqlmon.h</code> and is one of: <ul style="list-style-type: none"> • AUTORESIZE_LOCK • AUTOSTORAGE_LOCK • BLOCK_LOCK • EOT_LOCK • INPLACE_REORG_LOCK • INTERNAL_LOCK • INTERNALB_LOCK • INTERNALC_LOCK • INTERNALJ_LOCK • INTERNALL_LOCK • INTERNALO_LOCK • INTERNALQ_LOCK • INTERNALP_LOCK • INTERNALS_LOCK • INTERNALT_LOCK • INTERNALV_LOCK • KEYVALUE_LOCK • ROW_LOCK • SYSBOOT_LOCK • TABLE_LOCK • TABLE_PART_LOCK • TABLESPACE_LOCK • XML_PATH_LOCK |

Table 327. Information returned by the SNAPLOCKWAIT administrative view and the SNAP_GET_LOCKWAIT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|--------------|---|
| AGENT_ID_HOLDING_LK | BIGINT | agent_id_holding_lock - Agent ID holding lock |
| LOCK_WAIT_START_TIME | TIMESTAMP | lock_wait_start_time - Lock wait start timestamp |
| LOCK_MODE_REQUESTED | VARCHAR(10) | lock_mode_requested - Lock mode requested. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock escalation |
| TABNAME | VARCHAR(128) | table_name - Table name |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| APPL_ID_HOLDING_LK | VARCHAR(128) | appl_id_holding_lk - Application ID holding lock |
| LOCK_ATTRIBUTES | VARCHAR(128) | lock_attributes - Lock attributes. This interface returns a text identifier based on the defines in sqlmon.h. If there are no locks, the text identifier is NONE, otherwise, it is any combination of the following separated by a '+' sign: <ul style="list-style-type: none"> • ALLOW_NEW • DELETE_IN_BLOCK • ESCALATED • INSERT • NEW_REQUEST • RR • RR_IN_BLOCK • UPDATE_DELETE • WAIT_FOR_AVAIL |

Table 327. Information returned by the SNAPLOCKWAIT administrative view and the SNAP_GET_LOCKWAIT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|--|
| LOCK_CURRENT_MODE | VARCHAR(10) | lock_current_mode - Original lock mode before conversion. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • IN • IS • IX • NON (if no lock) • NS • NW • S • SIX • U • X • Z |
| LOCK_NAME | VARCHAR(32) | lock_name - Lock name |
| LOCK_RELEASE_FLAGS | BIGINT | lock_release_flags - Lock release flags. |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition identifier. For a non-partitioned table, this element is NULL. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPDB administrative view and SNAP_GET_DB table function - Retrieve snapshot information from the dbase logical group

The SNAPDB administrative view and the SNAP_GET_DB table function return snapshot information from the database (dbase) logical group.

Important: The SNAPDB administrative view and SNAP_GET_DB table function are deprecated and have been replaced by the “MON_GET_DATABASE table function - Get database metrics” on page 619 and “MON_GET_TRANSACTION_LOG table function - Get log information” on page 827.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPDB administrative view” on page 1308
- “SNAP_GET_DB table function” on page 1310

SNAPDB administrative view

This administrative view allows you to retrieve snapshot information from the dbase logical group for the currently connected database.

Used in conjunction with ADMIN_GET_STORAGE_PATHS table function, MON_GET_MEMORY_POOL, MON_GET_TRANSACTION_LOG, and MON_GET_HADR, the SNAPDB administrative view provides information equivalent to the **GET SNAPSHOT FOR DATABASE on database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 285 on page 1312 for a complete list of information that is returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPDB administrative view
- CONTROL privilege on the SNAPDB administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_DB table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Examples

Retrieve the status, platform, location, and connect time for all database members of the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME, DB_STATUS, SERVER_PLATFORM,
       DB_LOCATION, DB_CONN_TIME, DBPARTITIONNUM
FROM SYSIBMADM.SNAPDB ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| DB_NAME | DB_STATUS | SERVER_PLATFORM | DB_LOCATION | ... |
|-----------|-----------|-----------------|-------------|-----|
| -----...- | ----- | ----- | ----- | ... |
| TEST | ACTIVE | AIX64 | LOCAL | ... |

| | | | | |
|------|--------|-------|-------|-----|
| TEST | ACTIVE | AIX64 | LOCAL | ... |
| TEST | ACTIVE | AIX64 | LOCAL | ... |

3 record(s) selected.

Output from this query (continued).

| | | |
|-----|----------------------------|----------------|
| ... | DB_CONN_TIME | DBPARTITIONNUM |
| ... | ----- | ----- |
| ... | 2006-01-08-16.48.30.665477 | 0 |
| ... | 2006-01-08-16.48.34.005328 | 1 |
| ... | 2006-01-08-16.48.34.007937 | 2 |

This routine can be used by calling the following on the command line:

```

SELECT TOTAL_OLAP_FUNCS, OLAP_FUNC_OVERFLOWS, ACTIVE_OLAP_FUNCS
FROM SYSIBMADM.SNAPDB

```

| | | |
|------------------|---------------------|-------------------|
| TOTAL_OLAP_FUNCS | OLAP_FUNC_OVERFLOWS | ACTIVE_OLAP_FUNCS |
| ----- | ----- | ----- |
| | 7 | 2 |
| | | 1 |

1 record(s) selected.

After running a workload, a user can use the following query:

```

SELECT STATS_CACHE_SIZE, STATS_FABRICATIONS, SYNC_RUNSTATS,
ASYNC_RUNSTATS, STATS_FABRICATE_TIME, SYNC_RUNSTATS_TIME
FROM SYSIBMADM.SNAPDB

```

| | | | | |
|------------------|--------------------|---------------|----------------|-------|
| STATS_CACHE_SIZE | STATS_FABRICATIONS | SYNC_RUNSTATS | ASYNC_RUNSTATS | ... |
| ----- | ----- | ----- | ----- | ----- |
| 128 | 2 | 1 | 0 | ... |

| | | |
|-----|----------------------|--------------------|
| ... | STATS_FABRICATE_TIME | SYNC_RUNSTATS_TIME |
| ... | ----- | ----- |
| ... | 10 | 100 |

1 record(s) selected.

SNAP_GET_DB table function

The SNAP_GET_DB table function returns the same information as the SNAPDB administrative view.

Used in conjunction with the ADMIN_GET_STORAGE_PATHS table function, MON_GET_MEMORY_POOL, MON_GET_TRANSACTION_LOG, and MON_GET_HADR table functions, the SNAP_GET_DB table function provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES** CLP command.

Refer to Table 285 on page 1312 for a complete list of information that is returned.

Syntax

```

▶▶ SNAP_GET_DB ( ( dbname [ , member ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_DB table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_DB table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Retrieve the status, platform, location, and connect time as an aggregate view across all database members of the currently connected database.

```
SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME, DB_STATUS, SERVER_PLATFORM,  
       DB_LOCATION, DB_CONN_TIME FROM TABLE(SNAP_GET_DB('', -2)) AS T
```

The following is an example of output from this query.

```

DB_NAME      DB_STATUS      SERVER_PLATFORM ...
-----...-
SAMPLE      ACTIVE      AIX64      ...

1 record(s) selected.

```

Output from this query (continued).

```

... DB_LOCATION DB_CONN_TIME
... -----
... LOCAL      2005-07-24-22.09.22.013196

```

Example 2: Retrieve the status, platform, location, and connect time as an aggregate view across all database members for all active databases in the same instance that contains the currently connected database.

```

SELECT SUBSTR(DB_NAME, 1, 20) AS DB_NAME, DB_STATUS, SERVER_PLATFORM,
       DB_LOCATION, DB_CONN_TIME
FROM TABLE(SNAP_GET_DB(CAST (NULL AS VARCHAR(128)), -2)) AS T

```

The following is an example of output from this query.

```

DB_NAME      DB_STATUS      SERVER_PLATFORM ...
-----...-
TOOLSDB      ACTIVE      AIX64      ...
SAMPLE      ACTIVE      AIX64      ...

```

Output from this query (continued).

```

... DB_LOCATION DB_CONN_TIME
... -----
... LOCAL      2005-07-24-22.26.54.396335
... LOCAL      2005-07-24-22.09.22.013196

```

Example 3: This routine can be used by calling the following on the command line:

When connected to a database:

```

SELECT TOTAL_OLAP_FUNCS, OLAP_FUNC_OVERFLOWS, ACTIVE_OLAP_FUNCS
FROM TABLE (SNAP_GET_DB('', 0)) AS T

```

The output will look like:

```

TOTAL_OLAP_FUNCS  OLAP_FUNC_OVERFLOWS  ACTIVE_OLAP_FUNCS
-----
7                2                1

```

1 record(s) selected.

Example 4: After running a workload, a user can use the following query with the table function.

```

SELECT STATS_CACHE_SIZE, STATS_FABRICATIONS, SYNC_RUNSTATS,
       ASYNC_RUNSTATS, STATS_FABRICATE_TIME, SYNC_RUNSTATS_TIME
FROM TABLE (SNAP_GET_DB('mytestdb', -1)) AS SNAPDB

STATS_CACHE_SIZE  STATS_FABRICATIONS  SYNC_RUNSTATS  ASYNC_RUNSTATS ...
-----
200              1                2              0 ...

```

Continued

```

...STATS_FABRICATE_TIME  SYNC_RUNSTATS_TIME
...-----
...                2                32

```

1 record(s) selected.

Example 5: The following example shows how you can use the SNAP_GET_DB table function to determine the status of a database:

```
SELECT SUBSTR
      (DB_NAME, 1, 20) AS DB_NAME, DB_STATUS
FROM   table(SNAP_GET_DB('hadrdb', 0))
```

| | |
|---------|----------------|
| DB_NAME | DB_STATUS |
| ----- | ----- |
| HADRDB | ACTIVE_STANDBY |

Information returned

Table 328. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function

| Column name | Data type | Description or corresponding monitor element |
|------------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| DB_PATH | VARCHAR(1024) | db_path - Database path |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input database alias |
| DB_STATUS | VARCHAR(16) | db_status - Status of database. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • ACTIVE • QUIESCE_PEND • QUIESCED • ROLLFWD • ACTIVE_STANDBY - the HADR database is in a standby mode with reads on standby enabled. • STANDBY - the HADR database is in standby mode (reads on standby are not enabled). |
| CATALOG_PARTITION | SMALLINT | catalog_node - Catalog database partition number |
| CATALOG_PARTITION_NAME | VARCHAR(128) | catalog_node_name - Catalog database partition network name |

Table 328. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------|-------------|---|
| SERVER_PLATFORM | VARCHAR(12) | <p>server_platform - Server operating system. This interface returns a text identifier based on defines in sqlmon.h, and is one of:</p> <ul style="list-style-type: none"> • AIX • AIX64 • AS400_DRDA • DOS • DYNIX • HP • HP64 • HPIA • HPIA64 • LINUX • LINUX390 • LINUXIA64 • LINUXPPC • LINUXPPC64 • LINUXX8664 • LINUXZ64 • MAC • MVS_DRDA • NT • NT64 • OS2 • OS390 • SCO • SGI • SNI • SUN • SUN64 • UNKNOWN • UNKNOWN_DRDA • VM_DRDA • VSE_DRDA • WINDOWS |
| DB_LOCATION | VARCHAR(12) | <p>db_location - Database location. This interface returns a text identifier based on defines in sqlmon.h, and is one of:</p> <ul style="list-style-type: none"> • LOCAL • REMOTE |
| DB_CONN_TIME | TIMESTAMP | db_conn_time - Database activation timestamp |
| LAST_RESET | TIMESTAMP | last_reset - Last reset timestamp |
| LAST_BACKUP | TIMESTAMP | last_backup - Last backup timestamp |

Table 328. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| CONNECTIONS_TOP | BIGINT | connections_top - Maximum number of concurrent connections |
| TOTAL_CONS | BIGINT | total_cons - Connects since database activation |
| TOTAL_SEC_CONS | BIGINT | total_sec_cons - Secondary connections |
| APPLS_CUR_CONS | BIGINT | appls_cur_cons - Applications connected currently |
| APPLS_IN_DB2 | BIGINT | appls_in_db2 - Applications executing in the database currently |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of associated agents |
| AGENTS_TOP | BIGINT | agents_top - Number of agents created |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum number of coordinating agents |
| LOCKS_HELD | BIGINT | locks_held - Locks held |
| LOCK_WAITS | BIGINT | lock_waits - Lock waits |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time waited on locks |
| LOCK_LIST_IN_USE | BIGINT | lock_list_in_use - Total lock list memory in use |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks detected |
| LOCK_ESCALS | BIGINT | lock_escals - Number of lock escalations |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive lock escalations |
| LOCKS_WAITING | BIGINT | locks_waiting - Current agents waiting on locks |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of lock timeouts |
| NUM_INDOUBT_TRANS | BIGINT | num_indoubt_trans - Number of indoubt transactions |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total sort heap allocated |
| SORT_SHRHEAP_ALLOCATED | BIGINT | sort_shrheap_allocated - Sort share heap currently allocated |
| SORT_SHRHEAP_TOP | BIGINT | sort_shrheap_top - Sort share heap high water mark |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post shared threshold sorts |
| TOTAL_SORTS | BIGINT | total_sorts - Total sorts |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total sort time |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| ACTIVE_SORTS | BIGINT | active_sorts - Active sorts |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |

Table 328. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer pool asynchronous read time |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time |

Table 328. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|---|
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests |
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer pool no victim buffers |
| POOL_LSN_GAP_CLNS | BIGINT | pool_lsn_gap_clns - Buffer pool log space cleaners triggered |
| POOL_DRTY_PG_STEAL_CLNS | BIGINT | pool_drty_pg_steal_clns - Buffer pool victim page cleaners triggered |
| POOL_DRTY_PG_THRSH_CLNS | BIGINT | pool_drty_pg_thrsh_clns - Buffer pool threshold cleaners triggered |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time waited for prefetch |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| FILES_CLOSED | BIGINT | files_closed - Database files closed |
| ELAPSED_EXEC_TIME_S | BIGINT | elapsed_exec_time - Statement execution elapsed time |
| ELAPSED_EXEC_TIME_MS | BIGINT | elapsed_exec_time - Statement execution elapsed time |
| COMMIT_SQL_STMTS | BIGINT | commit_sql_stmts - Commit statements attempted |
| ROLLBACK_SQL_STMTS | BIGINT | rollback_sql_stmts - Rollback statements attempted |
| DYNAMIC_SQL_STMTS | BIGINT | dynamic_sql_stmts - Dynamic SQL statements attempted |
| STATIC_SQL_STMTS | BIGINT | static_sql_stmts - Static SQL statements attempted |
| FAILED_SQL_STMTS | BIGINT | failed_sql_stmts - Failed statement operations |
| SELECT_SQL_STMTS | BIGINT | select_sql_stmts - Select SQL statements executed |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - UPDATE/INSERT/DELETE SQL statements executed |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data definition language (DDL) SQL statements |
| INT_AUTO_REBINDS | BIGINT | int_auto_rebinds - Internal automatic rebinds |

Table 328. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|-----------|--|
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal rows deleted |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal rows inserted |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal rows updated |
| INT_COMMITS | BIGINT | int_commits - Internal commits |
| INT_ROLLBACKS | BIGINT | int_rollback - Internal rollbacks |
| INT_DEADLOCK_ROLLBACKS | BIGINT | int_deadlock_rollback - Internal rollbacks due to deadlock |
| ROWS_DELETED | BIGINT | rows_deleted - Rows deleted |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows inserted |
| ROWS_UPDATED | BIGINT | rows_updated - Rows updated |
| ROWS_SELECTED | BIGINT | rows_selected - Rows selected |
| ROWS_READ | BIGINT | rows_read - Rows read |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/precompiles attempted |
| TOTAL_LOG_AVAILABLE | BIGINT | total_log_available - Total log available |
| TOTAL_LOG_USED | BIGINT | total_log_used - Total log space used |
| SEC_LOG_USED_TOP | BIGINT | sec_log_used_top - Maximum secondary log space used |
| TOT_LOG_USED_TOP | BIGINT | tot_log_used_top - Maximum total log space used |
| SEC_LOGS_ALLOCATED | BIGINT | sec_logs_allocated - Secondary logs allocated currently |
| LOG_READS | BIGINT | log_reads - Number of log pages read |
| LOG_READ_TIME_S | BIGINT | log_read_time - Log read time |
| LOG_READ_TIME_NS | BIGINT | log_read_time - Log read time |
| LOG_WRITES | BIGINT | log_writes - Number of log pages written |
| LOG_WRITE_TIME_S | BIGINT | log_write_time - Log write time |
| LOG_WRITE_TIME_NS | BIGINT | log_write_time - Log write time |
| NUM_LOG_WRITE_IO | BIGINT | num_log_write_io - Number of log writes |
| NUM_LOG_READ_IO | BIGINT | num_log_read_io - Number of log reads |
| NUM_LOG_PART_PAGE_IO | BIGINT | num_log_part_page_io - Number of partial log page writes |
| NUM_LOG_BUFFER_FULL | BIGINT | num_log_buffer_full - Number of full log buffers |
| NUM_LOG_DATA_FOUND_IN_BUFFER | BIGINT | num_log_data_found_in_buffer - Number of log data found in buffer |
| APPL_ID_OLDEST_XACT | BIGINT | appl_id_oldest_xact - Application with oldest transaction |
| LOG_TO_REDO_FOR_RECOVERY | BIGINT | log_to_redo_for_recovery - Amount of log to be redone for recovery |
| LOG_HELD_BY_DIRTY_PAGES | BIGINT | log_held_by_dirty_pages - Amount of log space accounted for by dirty pages |

Table 328. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------------|-----------|--|
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package cache lookups |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package cache inserts |
| PKG_CACHE_NUM_OVERFLOWS | BIGINT | pkg_cache_num_overflows - Package cache overflows |
| PKG_CACHE_SIZE_TOP | BIGINT | pkg_cache_size_top - Package cache high water mark |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - Section lookups |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Section inserts |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog cache lookups |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog cache inserts |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog cache overflows |
| CAT_CACHE_SIZE_TOP | BIGINT | cat_cache_size_top - Catalog cache high water mark |
| PRIV_WORKSPACE_SIZE_TOP | BIGINT | priv_workspace_size_top - Maximum private workspace size |
| PRIV_WORKSPACE_NUM_OVERFLOWS | BIGINT | priv_workspace_num_overflows - Private workspace overflows |
| PRIV_WORKSPACE_SECTION_INSERTS | BIGINT | priv_workspace_section_inserts - Private workspace section inserts |
| PRIV_WORKSPACE_SECTION_LOOKUPS | BIGINT | priv_workspace_section_lookups - Private workspace section lookups |
| SHR_WORKSPACE_SIZE_TOP | BIGINT | shr_workspace_size_top - Maximum shared workspace size |
| SHR_WORKSPACE_NUM_OVERFLOWS | BIGINT | shr_workspace_num_overflows - Shared workspace overflows |
| SHR_WORKSPACE_SECTION_INSERTS | BIGINT | shr_workspace_section_inserts - Shared workspace section inserts |
| SHR_WORKSPACE_SECTION_LOOKUPS | BIGINT | shr_workspace_section_lookups - Shared workspace section lookups |
| TOTAL_HASH_JOINS | BIGINT | total_hash_joins - Total hash joins |
| TOTAL_HASH_LOOPS | BIGINT | total_hash_loops - Total hash loops |
| HASH_JOIN_OVERFLOWS | BIGINT | hash_join_overflows - Hash join overflows |
| HASH_JOIN_SMALL_OVERFLOWS | BIGINT | hash_join_small_overflows - Hash join small overflows |
| POST_SHRTHRESHOLD_HASH_JOINS | BIGINT | post_shrthreshold_hash_joins - Post threshold hash joins |
| ACTIVE_HASH_JOINS | BIGINT | active_hash_joins - Active hash joins |
| NUM_DB_STORAGE_PATHS | BIGINT | num_db_storage_paths - Number of automatic storage paths |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| SMALLEST_LOG_AVAIL_NODE | INTEGER | smallest_log_avail_node - Node with least available log space |

Table 328. Information returned by the SNAPDB administrative view and SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|-----------|--|
| TOTAL_OLAP_FUNCS | BIGINT | total_olap_funcs - Total OLAP functions |
| OLAP_FUNC_OVERFLOWS | BIGINT | olap_func_overflows - OLAP function overflows |
| ACTIVE_OLAP_FUNCS | BIGINT | active_olap_funcs - Active OLAP functions |
| STATS_CACHE_SIZE | BIGINT | stats_cache_size - Size of statistics cache |
| STATS_FABRICATIONS | BIGINT | stats_fabrications - Total number of statistics fabrications |
| SYNC_RUNSTATS | BIGINT | sync_runstats - Total number of synchronous RUNSTATS activities |
| ASYNCRUNSTATS | BIGINT | async_runstats - Total number of asynchronous RUNSTATS requests |
| STATS_FABRICATE_TIME | BIGINT | stats_fabricate_time - Total time spent on statistics fabrication activities |
| SYNC_RUNSTATS_TIME | BIGINT | sync_runstats_time - Total time spent on synchronous RUNSTATS activities |
| NUM_THRESHOLD_VIOLATIONS | BIGINT | num_threshold_violations - Number of threshold violations |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPUTIL administrative view and SNAP_GET_UTIL table function - Retrieve utility_info logical data group snapshot information

The SNAPUTIL administrative view and the SNAP_GET_UTIL table function return snapshot information about the utilities from the utility_info logical data group.

Important: The SNAPUTIL administrative view and the SNAP_GET_UTIL table function are deprecated and have been replaced by the “MON_GET_UTILITY table function - Get utilities running on the database” on page 863.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPUTIL administrative view” on page 1424
- “SNAP_GET_UTIL table function” on page 1426

SNAPUTIL administrative view

Used in conjunction with the SNAPUTIL_PROGRESS administrative view, the SNAPUTIL administrative view provides the same information as the **LIST UTILITIES SHOW DETAIL** CLP command.

The schema is SYSIBMADM.

Refer to Table 310 on page 1427 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPUTIL administrative view
- CONTROL privilege on the SNAPUTIL administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_UTIL table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of utilities and their states on all database members for all active databases in the instance that contains the connected database.

```
SELECT UTILITY_TYPE, UTILITY_PRIORITY, SUBSTR(UTILITY_DESCRIPTION, 1, 72)
      AS UTILITY_DESCRIPTION, SUBSTR(UTILITY_DBNAME, 1, 17) AS
      UTILITY_DBNAME, UTILITY_STATE, UTILITY_INVOKER_TYPE, DBPARTITIONNUM
FROM SYSIBMADM.SNAPUTIL ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| UTILITY_TYPE | UTILITY_PRIORITY | ... |
|--------------|------------------|-----|
| LOAD | - | ... |
| LOAD | - | ... |
| LOAD | - | ... |

3 record(s) selected.

Output from this query (continued).

| UTILITY_DESCRIPTION | ... |
|--|-----|
| ONLINE LOAD DEL AUTOMATIC INDEXING INSERT COPY NO TEST .LOADTEST | ... |
| ONLINE LOAD DEL AUTOMATIC INDEXING INSERT COPY NO TEST .LOADTEST | ... |
| ONLINE LOAD DEL AUTOMATIC INDEXING INSERT COPY NO TEST .LOADTEST | ... |

Output from this query (continued).

| ... UTILITY_DBNAME | UTILITY_STATE | UTILITY_INVOKER_TYPE | DBPARTITIONNUM |
|--------------------|---------------|----------------------|----------------|
| ... SAMPLE | EXECUTE | USER | 0 |
| ... SAMPLE | EXECUTE | USER | 1 |
| ... SAMPLE | EXECUTE | USER | 2 |

SNAP_GET_UTIL table function

The SNAP_GET_UTIL table function returns the same information as the SNAPUTIL administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

Used in conjunction with the SNAP_GET_UTIL_PROGRESS table function, the SNAP_GET_UTIL table function provides the same information as the **LIST UTILITIES SHOW DETAIL CLP** command.

Refer to Table 310 on page 1427 for a complete list of information that can be returned.

Syntax

```

>> SNAP_GET_UTIL ( ( member ) )

```

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_UTIL table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_UTIL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT

- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Retrieve a list of utility ids with their type and state for the currently connected database member on database SAMPLE.

```
SELECT UTILITY_ID, UTILITY_TYPE, UTILITY_STATE
FROM TABLE(SNAP_GET_UTIL(-1)) AS T WHERE UTILITY_DBNAME='SAMPLE'
```

The following is an example of output from this query:

| UTILITY_ID | UTILITY_TYPE | STATE |
|------------|--------------|---------|
| 1 | BACKUP | EXECUTE |

1 record(s) selected.

Example 2: Retrieve a list of utility ids with their type, member number and database partition number for the currently connected database member.

```
SELECT UTILITY_ID, UTILITY_TYPE, MEMBER, DBPARTITIONNUM
FROM TABLE(SNAP_GET_UTIL(-1)) AS T
```

The following is an example of output from this query:

| UTILITY_ID | UTILITY_TYPE | MEMBER | DBPARTITIONNUM |
|------------|--------------|--------|----------------|
| 2 | BACKUP | 2 | 2 |

Information returned

Table 329. Information returned by the SNAPUTIL administrative view and the SNAP_GET_UTIL table function

| Column name | Data type | Description or corresponding monitor element |
|---------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| UTILITY_ID | INTEGER | utility_id - Utility ID. Unique to a database partition. |
| UTILITY_TYPE | VARCHAR(26) | utility_type - Utility type. This interface returns a text identifier based on the defines in sqlmon.h. |
| UTILITY_PRIORITY | INTEGER | utility_priority - Utility priority. Priority if utility supports throttling, otherwise null. |
| UTILITY_DESCRIPTION | VARCHAR(2048) | utility_description - Utility description. Can be null. |
| UTILITY_DBNAME | VARCHAR(128) | utility_dbname - Database operated on by utility |
| UTILITY_START_TIME | TIMESTAMP | utility_start_time - Utility start time |

Table 329. Information returned by the SNAPUTIL administrative view and the SNAP_GET_UTIL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|--|
| UTILITY_STATE | VARCHAR(10) | utility_state - Utility state. This interface returns a text identifier based on the defines in sqlmon.h. |
| UTILITY_INVOKER_TYPE | VARCHAR(10) | utility_invoker_type - Utility invoker type. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> AUTO USER |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| PROGRESS_LIST_ATTR | VARCHAR(10) | progress_list_attr - Current progress list attributes |
| PROGRESS_LIST_CUR_SEQ_NUM | INTEGER | progress_list_current_seq_num - Current progress list sequence number |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPSTMT administrative view and SNAP_GET_STMT table function - Retrieve statement snapshot information

The SNAPSTMT administrative view and the SNAP_GET_STMT table function return information about SQL or XQuery statements from an application snapshot.

Important: The SNAPSTMT administrative view and the SNAP_GET_STMT table function are deprecated. Use the “MON_GET_ACTIVITY table function - Return a list of activities” on page 514 to retrieve the information returned by these deprecated interfaces.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPSTMT administrative view” on page 1382
- “SNAP_GET_STMT table function” on page 1383

SNAPSTMT administrative view

This administrative view allows you to retrieve statement snapshot information for the currently connected database.

Used with the MON_GET_AGENT, MON_GET_CONNECTION, MON_GET_MEMORY_POOL, and MON_GET_MEMORY_SET table functions and the MON_CURRENT_SQL and SNAPSUBSECTION administrative views, the SNAPSTMT administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS on database-alias** CLP command, but retrieves data from all database members.

The schema is SYSIBMADM.

Refer to Table 300 on page 1385 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPSTMT administrative view
- CONTROL privilege on the SNAPSTMT administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_STMT table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve rows read, written and operation performed for statements executed on the currently connected single-member database.

```
SELECT SUBSTR(STMT_TEXT,1,30) AS STMT_TEXT, ROWS_READ, ROWS_WRITTEN,  
       STMT_OPERATION FROM SYSIBMADM.SNAPSTMT
```

The following is an example of output from this query.

| STMT_TEXT | ROWS_READ | ROWS_WRITTEN | STMT_OPERATION |
|-----------|-----------|--------------|----------------|
| - | 0 | 0 | FETCH |
| - | 0 | 0 | STATIC_COMMIT |

2 record(s) selected.

SNAP_GET_STMT table function

The SNAP_GET_STMT table function returns the same information as the SNAPSTMT administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_AGENT, SNAP_GET_AGENT_MEMORY_POOL, SNAP_GET_APPL, SNAP_GET_APPL_INFO and SNAP_GET_SUBSECTION table functions, the SNAP_GET_STMT table function provides information equivalent to the **GET SNAPSHOT FOR ALL APPLICATIONS** CLP command, but retrieves data from all database partitions.

Refer to Table 300 on page 1385 for a complete list of information that can be returned.

Syntax

```

▶▶ SNAP_GET_STMT ( ( dbname [ , member ] ) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_STMT table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_STMT table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT

- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve rows read, written and operation performed for statements executed on current database member of currently connected database.

```
SELECT SUBSTR(STMT_TEXT,1,30) AS STMT_TEXT, ROWS_READ,
      ROWS_WRITTEN, STMT_OPERATION FROM TABLE(SNAP_GET_STMT('','-1)) AS T
```

The following is an example of output from this query.

```
STMT_TEXT                                ROWS_READ    ...
-----
update t set a=3                          0 ...
SELECT SUBSTR(STMT_TEXT,1,30)             0 ...
-                                           0 ...
-                                           0 ...
update t set a=2                          9 ...
5 record(s) selected.                      ...
```

Output from this query (continued).

```
... ROWS_WRITTEN  STMT_OPERATION
... -----
...              0 EXECUTE_IMMEDIATE
...              0 FETCH
...              0 NONE
...              0 NONE
...              1 EXECUTE_IMMEDIATE
...
```

Information returned

Table 330. Information returned by the SNAPSTMT administrative view and the SNAP_GET_STMT table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| AGENT_ID | BIGINT | agent_id - Application handle (agent ID) |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| NUM_AGENTS | BIGINT | num_agents - Number of agents working on a statement |
| AGENTS_TOP | BIGINT | agents_top - Number of agents created |

Table 330. Information returned by the SNAPSTMT administrative view and the SNAP_GET_STMT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|-------------|--|
| STMT_TYPE | VARCHAR(20) | stmt_type - Statement type. This interface returns a text identifier based on defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • DYNAMIC • NON_STMT • STATIC • STMT_TYPE_UNKNOWN |
| STMT_OPERATION | VARCHAR(20) | stmt_operation/operation - Statement operation. This interface returns a text identifier based on defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • CALL • CLOSE • COMPILE • DESCRIBE • EXECUTE • EXECUTE_IMMEDIATE • FETCH • FREE_LOCATOR • GETAA • GETNEXTCHUNK • GETTA • NONE • OPEN • PREP_COMMIT • PREP_EXEC • PREP_OPEN • PREPARE • REBIND • REDIST • REORG • RUNSTATS • SELECT • SET • STATIC_COMMIT • STATIC_ROLLBACK |
| SECTION_NUMBER | BIGINT | section_number - Section number |
| QUERY_COST_ESTIMATE | BIGINT | query_cost_estimate - Query cost estimate |
| QUERY_CARD_ESTIMATE | BIGINT | query_card_estimate - Query number of rows estimate |
| DEGREE_PARALLELISM | BIGINT | degree_parallelism - Degree of parallelism |
| STMT_SORTS | BIGINT | stmt_sorts - Statement sorts |

Table 330. Information returned by the SNAPSTMT administrative view and the SNAP_GET_STMT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|--------------|---|
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total sort time |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort overflows |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal rows deleted |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal rows updated |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal rows inserted |
| FETCH_COUNT | BIGINT | fetch_count - Number of successful fetches |
| STMT_START | TIMESTAMP | stmt_start - Statement operation start timestamp |
| STMT_STOP | TIMESTAMP | stmt_stop - Statement operation stop timestamp |
| STMT_USR_CPU_TIME_S | BIGINT | stmt_usr_cpu_time - User CPU time used by statement (in seconds)* |
| STMT_USR_CPU_TIME_MS | BIGINT | stmt_usr_cpu_time - User CPU time used by statement (fractional, in microseconds)* |
| STMT_SYS_CPU_TIME_S | BIGINT | stmt_sys_cpu_time - System CPU time used by statement (in seconds)* |
| STMT_SYS_CPU_TIME_MS | BIGINT | stmt_sys_cpu_time - System CPU time used by statement (fractional, in microseconds)* |
| STMT_ELAPSED_TIME_S | BIGINT | stmt_elapsed_time - Most recent statement elapsed time (in seconds)* |
| STMT_ELAPSED_TIME_MS | BIGINT | stmt_elapsed_time - Most recent statement elapsed time (fractional, in microseconds)* |
| BLOCKING_CURSOR | SMALLINT | blocking_cursor - Blocking cursor |
| STMT_NODE_NUMBER | SMALLINT | stmt_node_number - Statement node |
| CURSOR_NAME | VARCHAR(128) | cursor_name - Cursor name |
| CREATOR | VARCHAR(128) | creator - Application creator |
| PACKAGE_NAME | VARCHAR(128) | package_name - Package name |
| STMT_TEXT | CLOB(16 M) | stmt_text - SQL statement text |
| CONSISTENCY_TOKEN | VARCHAR(128) | consistency_token - Package consistency token |
| PACKAGE_VERSION_ID | VARCHAR(128) | package_version_id - Package version |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |

Table 330. Information returned by the SNAPSTMT administrative view and the SNAP_GET_STMT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--|-----------|---|
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads monitor element |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| <p>* To calculate the total time spent for the monitor element that this column is based on, you must add the full seconds reported in the column for this monitor element that ends with _S to the fractional seconds reported in the column for this monitor element that ends with _MS, using the following formula: $(\text{monitor-element-name_S} \times 1,000,000 + \text{monitor-element-name_MS}) \div 1,000,000$. For example, $(\text{ELAPSED_EXEC_TIME_S} \times 1,000,000 + \text{ELAPSED_EXEC_TIME_MS}) \div 1,000,000$.</p> | | |

SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS_V97 table function - Retrieve automatic storage path information

The SNAPSTORAGE_PATHS administrative view and the SNAP_GET_STORAGE_PATHS_V97 return a list of automatic storage paths for the database including file system information for each storage path, specifically, from the db_storage_group logical data group.

Note: The “SNAPSTORAGE_PATHS administrative view” on page 1390 and the “SNAP_GET_STORAGE_PATHS_V97” on page 1391 have been deprecated and replaced by the “ADMIN_GET_STORAGE_PATHS table function - retrieve automatic storage path information” on page 217. This function and view might be removed in a future release.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPSTORAGE_PATHS administrative view” on page 1390
- “SNAP_GET_STORAGE_PATHS_V97” on page 1391

SNAPSTORAGE_PATHS administrative view

This administrative view allows you to retrieve automatic storage path information for the currently connected database.

Used with the MON_GET_DATABASE, MON_GET_HADR, MON_GET_MEMORY_POOL, MON_GET_MEMORY_SET, and MON_GET_TRANSACTION_LOG table functions, the SNAPSTORAGE_PATHS administrative view provides information equivalent to the **GET SNAPSHOT FOR DATABASE ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 302 on page 1392 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPSTORAGE_PATHS administrative view
- CONTROL privilege on the SNAPSTORAGE_PATHS administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following authorizations is required to use the table function:

- EXECUTE privilege on the SNAP_GET_STORAGE_PATHS_V97 table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

To access snapshot monitor data, one of the following authorities is also required:

- SYSMON authority
- SYSCtrl authority
- SYSMAINT authority
- SYSADM authority

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve the storage path for the currently connected single-member database.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, SUBSTR(DB_STORAGE_PATH,1,8)
      AS DB_STORAGE_PATH, SUBSTR(HOSTNAME,1,10) AS HOSTNAME
FROM SYSIBMADM.SNAPSTORAGE_PATHS
```

The following is an example of output from this query.

| DB_NAME | DB_STORAGE_PATH | HOSTNAME |
|---------|-----------------|----------|
| STOPATH | d: | JESSICAE |

1 record(s) selected.

SNAP_GET_STORAGE_PATHS_V97

The SNAP_GET_STORAGE_PATHS_V97 table function returns similar information as the SNAPSTORAGE_PATHS administrative view. It allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_DB, SNAP_GET_DETAILLOG, SNAP_GET_HADR and SNAP_GET_DB_MEMORY_POOL table functions, the SNAP_GET_STORAGE_PATHS_V97 table function provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES** CLP command.

Refer to Table 302 on page 1392 for a complete list of information that can be returned.

Syntax

►►—SNAP_GET_STORAGE_PATHS_V97—(—*dbname*— — —*member*—)

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify an empty string to take the snapshot from the currently connected database. Specify a NULL value to take the snapshot from all databases within the same instance as the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid member number. Specify -1 for the current member, or -2 for an aggregate of all active members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all members where the database is active.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file

with the corresponding snapshot API request type does not exist, then the SNAP_GET_STORAGE_PATHS_V97 table function takes a snapshot for the currently connected database and database member.

Authorization

- EXECUTE privilege on the SNAP_GET_STORAGE_PATHS_V97 table function.
- DATAACCESS authority

To access snapshot monitor data, one of the following authorities is required:

- SYSMON authority
- SYSCTRL authority
- SYSMAINT authority
- SYSADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Retrieve the storage path information for all active databases.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, DB_STORAGE_PATH
FROM TABLE(SNAP_GET_STORAGE_PATHS_V97(CAST (NULL AS VARCHAR(128)), -1)) AS T
```

The following is an example of output from this query.

```
DB_NAME  DB_STORAGE_PATH
-----  -
STOPATH  /home/jessica/sdb
MYDB     /home/jessica/mdb

2 record(s) selected
```

Information returned

The BUFFERPOOL monitor switch must be turned on in order for the file system information to be returned.

Table 331. Information returned by the SNAPSTORAGE_PATHS administrative view and the SNAP_GET_STORAGE_PATHS_V97 table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| DB_NAME | VARCHAR(128) | db_name - Database name |
| DB_STORAGE_PATH | VARCHAR(256) | db_storage_path - Automatic storage path |
| DB_STORAGE_PATH_WITH_DPE | VARCHAR(256) | db_storage_path_with_dpe - Storage path including database partition expression monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

Table 331. Information returned by the `SNAPSTORAGE_PATHS` administrative view and the `SNAP_GET_STORAGE_PATHS_V97` table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|-------------|--|
| DB_STORAGE_PATH_STATE | VARCHAR(16) | db_storage_path_state - Storage path state monitor element |
| FS_ID | VARCHAR(22) | fs_id - Unique file system identification number |
| FS_TOTAL_SIZE | BIGINT | fs_total_size - Total size of a file system |
| FS_USED_SIZE | BIGINT | fs_used_size - Amount of space used on a file system |
| STO_PATH_FREE_SIZE | BIGINT | sto_path_free_size - Automatic storage path free space |

SNAPTAB administrative view and SNAP_GET_TAB table function - Retrieve table logical data group snapshot information

The SNAPTAB administrative view and the SNAP_GET_TAB table function return snapshot information from the table logical data group.

Important: The SNAPTAB administrative view and SNAP_GET_TAB table function are deprecated. Use the “MON_GET_TABLE table function - get table metrics” on page 800, to retrieve the information returned by these deprecated interfaces.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPTAB administrative view” on page 1393
- “SNAP_GET_TAB table function” on page 1394

SNAPTAB administrative view

This administrative view allows you to retrieve table logical data group snapshot information for the currently connected database.

Used in conjunction with the SNAPTAB_REORG administrative view, the SNAPTAB administrative view returns equivalent information to the **GET SNAPSHOT FOR TABLES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 303 on page 1396 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPTAB administrative view
- CONTROL privilege on the SNAPTAB administrative view
- DATAACCESS authority

- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_TAB table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve the schema and name for all active tables.

```
SELECT SUBSTR(TABSCHEMA,1,8), SUBSTR(TABNAME,1,15) AS TABNAME, TAB_TYPE,
       DBPARTITIONNUM FROM SYSIBMADM.SNAPTAB
```

The following is an example of output from this query.

| TABSCHEMA | TABNAME | TAB_TYPE | DBPARTITIONNUM |
|-----------|---------------|------------|----------------|
| SYSTOOLS | HMON_ATM_INFO | USER_TABLE | 0 |

1 record selected.

SNAP_GET_TAB table function

The SNAP_GET_TAB table function returns the same information as the SNAPTAB administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used in conjunction with the SNAP_GET_TAB_REORG table function, the SNAP_GET_TAB table function returns equivalent information to the **GET SNAPSHOT FOR TABLES ON database-alias** CLP command.

Refer to Table 303 on page 1396 for a complete list of information that can be returned.

Syntax

```
►► SNAP_GET_TAB (—dbname— [ , member ] ) ►►
```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current member, or -2 for an aggregate of all active members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all members where the database is active.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TAB table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_TAB table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of active tables as an aggregate view for the currently connected database.

```
SELECT SUBSTR(TABSCHEMA,1,8) AS TABSCHEMA, SUBSTR(TABNAME,1,15) AS TABNAME,  
       TAB_TYPE, DBPARTITIONNUM FROM TABLE(SNAP_GET_TAB('','-2')) AS T
```

The following is an example of output from this query.

| TABSCHEMA | TABNAME | TAB_TYPE | DBPARTITIONNUM |
|-----------|---------------|------------|----------------|
| SYSTOOLS | HMON_ATM_INFO | USER_TABLE | - |
| JESSICAE | EMPLOYEE | USER_TABLE | - |

Information returned

Table 332. Information returned by the SNAPTAB administrative view and the SNAP_GET_TAB table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABSCHEMA | VARCHAR(128) | table_schema - Table schema name |
| TABNAME | VARCHAR(128) | table_name - Table name |
| TAB_FILE_ID | BIGINT | table_file_id - Table file identification |
| TAB_TYPE | VARCHAR(14) | table_type - Table type. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> CATALOG_TABLE DROPPED_TABLE REORG_TABLE SYNOPSIS_TABLE TEMP_TABLE USER_TABLE |
| DATA_OBJECT_PAGES | BIGINT | data_object_pages - Data object pages |
| INDEX_OBJECT_PAGES | BIGINT | index_object_pages - Index object pages |
| LOB_OBJECT_PAGES | BIGINT | lob_object_pages - LOB object pages |
| LONG_OBJECT_PAGES | BIGINT | long_object_pages - Long object pages |
| XDA_OBJECT_PAGES | BIGINT | xda_object_pages - XDA Object Pages |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_WRITTEN | BIGINT | rows_written - Rows written |
| OVERFLOW_ACCESSES | BIGINT | overflow_accesses - Accesses to overflowed records |
| PAGE_REORGS | BIGINT | page_reorgs - Page reorganizations |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition identifier. For a non-partitioned table, this element will be NULL. |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPTBSP administrative view and SNAP_GET_TBSP table function - Retrieve table space logical data group snapshot information

The SNAPTBSP administrative view and the SNAP_GET_TBSP table function return snapshot information from the table space logical data group.

Important: The SNAPTBSP administrative view and the SNAP_GET_TBSP table function are deprecated and have been replaced by the “MON_GET_TABLESPACE table function - Get table space metrics” on page 805.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPTBSP administrative view” on page 1400
- “SNAP_GET_TBSP table function” on page 1401

SNAPTBSP administrative view

This administrative view allows you to retrieve table space logical data group snapshot information for the currently connected database.

Used in conjunction with the SNAPTBSP_PART, SNAPTBSP_QUIESCER, SNAPTBSP_RANGE, SNAPCONTAINER administrative views, the SNAPTBSP administrative view returns information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 305 on page 1402 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPTBSP administrative view
- CONTROL privilege on the SNAPTBSP administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_TBSP table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of table spaces on the catalog database member for the currently connected database.

```
SELECT SUBSTR(TBSP_NAME,1,30) AS TBSP_NAME, TBSP_ID, TBSP_TYPE,
       TBSP_CONTENT_TYPE FROM SYSIBMADM.SNAPTBSPP WHERE DBPARTITIONNUM = 1
```

The following is an example of output from this query.

| TBSP_NAME | TBSP_ID | TBSP_TYPE | TBSP_CONTENT_TYPE |
|------------|---------|-----------|-------------------|
| TEMPSPACE1 | 1 | SMS | SYSTEMP |
| USERSPACE1 | 2 | DMS | LONG |

2 record(s) selected.

SNAP_GET_TBSP table function

The SNAP_GET_TBSP table function returns the same information as the SNAPTBSPP administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used in conjunction with the SNAP_GET_TBSP_PART, SNAP_GET_TBSP_QUIESCER, SNAP_GET_TBSP_RANGE, SNAP_GET_CONTAINER table functions, the SNAP_GET_TBSP table function returns information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 305 on page 1402 for a complete list of information that can be returned.

Syntax

```
→ SNAP_GET_TBSP ( (dbname [ , member ] ) ) →
```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An input argument of type INTEGER that specifies a valid member number. Specify -1 for the current member. If the null value is specified, -1 is set implicitly.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TBSP table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_TBSP table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of table spaces for all database members for the currently connected database.

```
SELECT SUBSTR(TBSP_NAME,1,10) AS TBSP_NAME, TBSP_ID, TBSP_TYPE,
       TBSP_CONTENT_TYPE, DBPARTITIONNUM FROM TABLE(SNAP_GET_TBSP('')) AS T
```

The following is an example of output from this query.

| TBSP_NAME | TBSP_ID | TBSP_TYPE | TBSP_CONTENT_TYPE | DBPARTITIONNUM |
|------------|---------|-----------|-------------------|----------------|
| TEMPSPACE1 | 1 | SMS | SYSTEMP | 1 |
| USERSPACE1 | 2 | DMS | LONG | 1 |
| SYSCATSPAC | 0 | DMS | ANY | 0 |
| TEMPSPACE1 | 1 | SMS | SYSTEMP | 0 |
| USERSPACE1 | 2 | DMS | LONG | 0 |
| SYSTOOLSPA | 3 | DMS | LONG | 0 |
| TEMPSPACE1 | 1 | SMS | SYSTEMP | 2 |
| USERSPACE1 | 2 | DMS | LONG | 2 |

8 record(s) selected.

Information returned

Table 333. Information returned by the SNAPTBSP administrative view and the SNAP_GET_TBSP table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |

Table 333. Information returned by the *SNAPTbsp* administrative view and the *SNAP_GET_Tbsp* table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|--------------|---|
| Tbsp_NAME | VARCHAR(128) | tablespace_name - Table space name |
| Tbsp_ID | BIGINT | tablespace_id - Table space identification |
| Tbsp_TYPE | VARCHAR(10) | tablespace_type - Table space type. This interface returns a text identifier based on defines in <code>sqlutil.h</code> , and is one of: <ul style="list-style-type: none"> • DMS • SMS |
| Tbsp_CONTENT_TYPE | VARCHAR(10) | tablespace_content_type - Table space contents type. This interface returns a text identifier based on defines in <code>sqlmon.h</code> , and is one of: <ul style="list-style-type: none"> • ANY • LARGE • SYSTEMP • USRTEMP |
| Tbsp_PAGE_SIZE | BIGINT | tablespace_page_size - Table space page size |
| Tbsp_EXTENT_SIZE | BIGINT | tablespace_extent_size - Table space extent size |
| Tbsp_PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table space prefetch size |
| Tbsp_CUR_POOL_ID | BIGINT | tablespace_cur_pool_id - Buffer pool currently being used |
| Tbsp_NEXT_POOL_ID | BIGINT | tablespace_next_pool_id - Buffer pool that will be used at next startup |
| FS_CACHING | SMALLINT | fs_caching - File system caching |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer pool data logical reads |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer pool data physical reads |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer pool temporary data logical reads |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer pool temporary data physical reads |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer pool asynchronous data reads |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer pool data writes |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer pool asynchronous data writes |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer pool index logical reads |

Table 333. Information returned by the *SNAPTbsp* administrative view and the *SNAP_GET_TBSP* table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|---|
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer pool index physical reads |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer pool temporary index logical reads |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer pool temporary index physical reads |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer pool asynchronous index reads |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer pool index writes |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer pool asynchronous index writes |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_READ_TIME | BIGINT | pool_read_time - Total buffer pool physical read time |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total buffer pool physical write time |
| POOL_ASYNC_READ_TIME | BIGINT | pool_async_read_time - Buffer pool asynchronous read time |
| POOL_ASYNC_WRITE_TIME | BIGINT | pool_async_write_time - Buffer pool asynchronous write time |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer pool asynchronous read requests |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer pool asynchronous index read requests |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests |

Table 333. Information returned by the SNAPTBSP administrative view and the SNAP_GET_TBSP table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|-------------|--|
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer pool no victim buffers |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |
| DIRECT_READ_REQS | BIGINT | direct_read_reqs - Direct read requests |
| DIRECT_WRITE_REQS | BIGINT | direct_write_reqs - Direct write requests |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct read time |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct write time |
| FILES_CLOSED | BIGINT | files_closed - Database files closed |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread prefetch pages |
| TBSP_REBALANCER_MODE | VARCHAR(10) | tablespace_rebalancer_mode - Rebalancer mode. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • NO_REBAL • FWD_REBAL • REV_REBAL |
| TBSP_USING_AUTO_STORAGE | SMALLINT | tablespace_using_auto_storage - Table space enabled for automatic storage |
| TBSP_AUTO_RESIZE_ENABLED | SMALLINT | tablespace_auto_resize_enabled - Table space automatic resizing enabled |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPTBSP_PART administrative view and SNAP_GET_TBSP_PART table function - Retrieve tablespace_nodeinfo logical data group snapshot information

The SNAPTBSP_PART administrative view and the SNAP_GET_TBSP_PART table function return snapshot information from the tablespace_nodeinfo logical data group.

Important: The SNAPTBSP_PART administrative view and the SNAP_GET_TBSP_PART table function are deprecated and have been replaced by the “MON_GET_TABLESPACE table function - Get table space metrics” on page 805.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPTBSP_PART administrative view” on page 1406
- “SNAP_GET_TBSP_PART table function” on page 1407

SNAPTBSP_PART administrative view

This administrative view allows you to retrieve tablespace_nodeinfo logical data group snapshot information for the currently connected database.

Used in conjunction with the SNAPTBSP, SNAPTBSP_QUIESCER, SNAPTBSP_RANGE, SNAPCONTAINER administrative views, the SNAPTBSP_PART administrative view returns information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 306 on page 1408 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPTBSP_RANGE administrative view
- CONTROL privilege on the SNAPTBSP_RANGE administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following authorizations is required to use the table function:

- EXECUTE privilege on the SNAP_GET_TBSP_PART table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMOINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of table spaces and their state for all database partitions of the currently connected database.

```
SELECT SUBSTR(TBSP_NAME,1,30) AS TBSP_NAME, TBSP_ID,
       SUBSTR(TBSP_STATE,1,30) AS TBSP_STATE, DBPARTITIONNUM
FROM SYSIBMADM.SNAPTbsp_PART
```

The following is an example of output from this query.

| TBSP_NAME | TBSP_ID | TBSP_STATE | DBPARTITIONNUM |
|-------------|---------|------------|----------------|
| SYSCATSPACE | 0 | NORMAL | 0 |
| TEMPSPACE1 | 1 | NORMAL | 0 |
| USERSPACE1 | 2 | NORMAL | 0 |
| TEMPSPACE1 | 1 | NORMAL | 1 |
| USERSPACE1 | 2 | NORMAL | 1 |

5 record(s) selected.

SNAP_GET_TBSP_PART table function

The SNAP_GET_TBSP_PART table function returns the same information as the SNAPTbsp_PART administrative view, but allows you to retrieve the information for a specific database on a specific database partition, aggregate of all database partitions or all database partitions.

Used in conjunction with the SNAP_GET_TBSP, SNAP_GET_TBSP QUIESCER, SNAP_GET_TBSP_RANGE, SNAP_GET_CONTAINER table functions, the SNAP_GET_TBSP_PART table function returns information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 306 on page 1408 for a complete list of information that can be returned.

Syntax

```
▶▶ SNAP_GET_TBSP_PART ( (—dbname— [ , member ] ) ) ▶▶
```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid member number. Specify -1 for the current member, or -2 for an aggregate of all active members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all members where the database is active.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the

SNAP_GET_TBSP_PART table function takes a snapshot for the currently connected database and member.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_TBSP_PART table function.
- DATAACCESS authority

To access snapshot monitor data, one of the following authorities is also required:

- SYSMON authority
- SYSCTRL authority
- SYSMANT authority
- SYSADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve a list of table spaces and their state for the connected database partition of the connected database.

```
SELECT SUBSTR(TBSP_NAME,1,30) AS TBSP_NAME, TBSP_ID,
       SUBSTR(TBSP_STATE,1,30) AS TBSP_STATE
FROM TABLE(SNAP_GET_TBSP_PART(CAST(NULL AS VARCHAR(128)),-1)) AS T
```

The following is an example of output from this query.

| TBSP_NAME | TBSP_ID | TBSP_STATE |
|------------------|---------|------------|
| ----- | ----- | ----- |
| SYSCATSPACE | | 0 NORMAL |
| TEMPSPACE1 | | 1 NORMAL |
| USERSPACE1 | | 2 NORMAL |
| SYSTOOLSPACE | | 3 NORMAL |
| SYSTOOLSTMPSPACE | | 4 NORMAL |

5 record(s) selected.

Information returned

Table 334. Information returned by the SNAPTbsp_PART administrative view and the SNAP_GET_TBSP_PART table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|---------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_NAME | VARCHAR (128) | tablespace_name - Table space name |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |

Table 334. Information returned by the SNAPTBSP_PART administrative view and the SNAP_GET_TBSP_PART table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------------|------------------|---|
| TBSP_STATE | VARCHAR
(256) | <p>tablespace_state - Table space state. This interface returns a text identifier based on defines in sqlutil.h and is combination of the following separated by a '+' sign:</p> <ul style="list-style-type: none"> • BACKUP_IN_PROGRESS • BACKUP_PENDING • DELETE_PENDING • DISABLE_PENDING • DROP_PENDING • LOAD_IN_PROGRESS • LOAD_PENDING • NORMAL • OFFLINE • PSTAT_CREATION • PSTAT_DELETION • QUIESCED_EXCLUSIVE • QUIESCED_SHARE • QUIESCED_UPDATE • REBAL_IN_PROGRESS • REORG_IN_PROGRESS • RESTORE_IN_PROGRESS • RESTORE_PENDING • ROLLFORWARD_IN_PROGRESS • ROLLFORWARD_PENDING • STORDEF_ALLOWED • STORDEF_CHANGED • STORDEF_FINAL_VERSION • STORDEF_PENDING • SUSPEND_WRITE |
| TBSP_PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table space prefetch size |
| TBSP_NUM_QUIESCERS | BIGINT | tablespace_num_quiescers - Number of quiescers |
| TBSP_STATE_CHANGE_OBJECT_ID | BIGINT | tablespace_state_change_object_id - State change object identification |
| TBSP_STATE_CHANGE_TBSP_ID | BIGINT | tablespace_state_change_ts_id - State change table space identification |
| TBSP_MIN_RECOVERY_TIME | TIMESTAMP | tablespace_min_recovery_time - Minimum recovery time for rollforward |
| TBSP_TOTAL_PAGES | BIGINT | tablespace_total_pages - Total pages in table space |
| TBSP_USABLE_PAGES | BIGINT | tablespace_usable_pages - Usable pages in table space |
| TBSP_USED_PAGES | BIGINT | tablespace_used_pages - Used pages in table space |

Table 334. Information returned by the *SNAPTbsp_PART* administrative view and the *SNAP_GET_Tbsp_PART* table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|--------------|--|
| TBSP_FREE_PAGES | BIGINT | tablespace_free_pages - Free pages in table space |
| TBSP_PENDING_FREE_PAGES | BIGINT | tablespace_pending_free_pages - Pending free pages in table space |
| TBSP_PAGE_TOP | BIGINT | tablespace_page_top - Table space high water mark |
| REBALANCER_MODE | VARCHAR (30) | tablespace_rebalancer_mode - Rebalancer mode. This interface returns a text identifier based on defines in <i>sqlmon.h</i> , and is one of: <ul style="list-style-type: none"> • FWD_REBAL • NO_REBAL • REV_REBAL • FWD_REBAL_OF_2PASS • REV_REBAL_OF_2PASS |
| REBALANCER_EXTENTS_REMAINING | BIGINT | tablespace_rebalancer_extents_remaining - Total number of extents to be processed by the rebalancer |
| REBALANCER_EXTENTS_PROCESSED | BIGINT | tablespace_rebalancer_extents_processed - Number of extents the rebalancer has processed |
| REBALANCER_PRIORITY | BIGINT | tablespace_rebalancer_priority - Current rebalancer priority |
| REBALANCER_START_TIME | TIMESTAMP | tablespace_rebalancer_start_time - Rebalancer start time |
| REBALANCER_RESTART_TIME | TIMESTAMP | tablespace_rebalancer_restart_time - Rebalancer restart time |
| REBALANCER_LAST_EXTENT_MOVED | BIGINT | tablespace_rebalancer_last_extent_moved - Last extent moved by the rebalancer |
| TBSP_NUM_RANGES | BIGINT | tablespace_num_ranges - Number of ranges in the table space map |
| TBSP_NUM_CONTAINERS | BIGINT | tablespace_num_containers - Number of containers in table space |
| TBSP_INITIAL_SIZE | BIGINT | tablespace_initial_size - Initial table space size |
| TBSP_CURRENT_SIZE | BIGINT | tablespace_current_size - Current table space size |
| TBSP_MAX_SIZE | BIGINT | tablespace_max_size - Maximum table space size |
| TBSP_INCREASE_SIZE | BIGINT | tablespace_increase_size - Increase size in bytes |
| TBSP_INCREASE_SIZE_PERCENT | SMALLINT | tablespace_increase_size_percent - Increase size by percent |
| TBSP_LAST_RESIZE_TIME | TIMESTAMP | tablespace_last_resize_time - Time of last successful resize |
| TBSP_LAST_RESIZE_FAILED | SMALLINT | tablespace_last_resize_failed - Last resize attempt failed |

Table 334. Information returned by the `SNAPTbsp_Part` administrative view and the `SNAP_Get_Tbsp_Part` table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| TBSP_PATHS_DROPPED | SMALLINT | Indicates that the table space resides on one or more storage paths that have been dropped (0 - No, 1 - Yes) |

SNAPTbsp_QUIESCER administrative view and SNAP_Get_Tbsp_QUIESCER table function - Retrieve quiescer table space snapshot information

The `SNAPTbsp_QUIESCER` administrative view and the `SNAP_Get_Tbsp_QUIESCER` table function return information about quiescers from a table space snapshot.

Important: The `SNAPTbsp_QUIESCER` administrative view and `SNAP_Get_Tbsp_QUIESCER` table function are deprecated and have been replaced by “`MON_Get_Tablespace_QUIESCER` table function - Get information about quiesced table spaces” on page 817.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “`SNAPTbsp_QUIESCER` administrative view” on page 1420
- “`SNAP_Get_Tbsp_QUIESCER` table function” on page 1422

SNAPTbsp_QUIESCER administrative view

This administrative view allows you to retrieve quiescer table space snapshot information for the currently connected database.

Used with the `SNAPTbsp_Range` administrative view and the `MON_Get_Tablespace` and `MON_Get_Container` table functions, the `SNAPTbsp_QUIESCER` administrative view provides information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

The schema is `SYSIBMADM`.

Refer to Table 309 on page 1423 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- `SELECT` privilege on the `SNAPTbsp_QUIESCER` administrative view
- `CONTROL` privilege on the `SNAPTbsp_QUIESCER` administrative view
- `DATAACCESS` authority
- `DBADM` authority
- `SQLADM` authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_TBSP QUIESCER table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve information about the quiesced table spaces for all database members for the currently connected database.

```
SELECT SUBSTR(TBSP_NAME, 1, 10) AS TBSP_NAME, QUIESCER_TS_ID,
       QUIESCER_OBJ_ID, QUIESCER_AUTH_ID, QUIESCER_AGENT_ID,
       QUIESCER_STATE, DBPARTITIONNUM
FROM SYSIBMADM.SNAPTBSPTQUIESCER ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| TBSP_NAME | QUIESCER_TS_ID | QUIESCER_OBJ_ID | QUIESCER_AUTH_ID | .. |
|------------|----------------|-----------------|------------------|----|
| USERSPACE1 | 2 | 5 | SWALKTY | .. |
| USERSPACE1 | 2 | 5 | SWALKTY | .. |

2 record(s) selected.

Output from this query (continued).

| QUIESCER_AGENT_ID | QUIESCER_STATE | DBPARTITIONNUM |
|-------------------|----------------|----------------|
| 0 | EXCLUSIVE | 0 |
| 65983 | EXCLUSIVE | 1 |

Example: Determine the partitioned table names

If the table is partitioned and kept in quiesced state, the different values for table space ID and table ID are represented than in SYSCAT.TABLES. These IDs will appear as the unsigned short representation. In order to find the quiesced table name, you need to find the signed short representation first by calculating the table space ID that is subtracting 65536 (the maximum value) from QUIESCER_TS_ID and then use this table space ID to locate the quiesced tables. (The actual table space ID can be found in SYSCAT.DATAPARTITIONS for each data partition in the table.)

```
SELECT SUBSTR(TBSP_NAME, 1, 10) AS TBSP_NAME,
       CASE WHEN QUIESCER_TS_ID = 65530
            THEN QUIESCER_TS_ID - 65536
            ELSE QUIESCER_TS_ID END as tbspaceid,
       CASE WHEN QUIESCER_TS_ID = 65530
```

```

        THEN QUIESCER_OBJ_ID - 65536
        ELSE QUIESCER_OBJ_ID END as tableid
FROM SYSIBMADM.SNAPTbsp_QUIESCER
ORDER BY DBPARTITIONNUM

```

The following is an example of output from this query.

| TBSP_NAME | TBSPACEID | TABLEID |
|-----------|-----------|---------|
| TABDATA | -6 | -32768 |
| DATAMART | -6 | -32765 |
| SMALL | 5 | 17 |

3 record(s) selected.

Use the given TBSPACEID and TABLEID provided from this query to find the table schema and name from SYSCAT.TABLES.

```

SELECT CHAR(tabschema, 10)tabschema, CHAR(tabname,15)tabname
FROM SYSCAT.TABLES
WHERE tbspaceid = -6 AND tableid in (-32768,-32765)

```

The following is an example of output from this query.

| TABSCHEMA | TABNAME |
|-----------|--------------|
| SAMP | ORDERS_RP |
| SAMP | ORDERS_DMART |

2 record(s) selected.

```

SELECT CHAR(tabschema, 10)tabschema, CHAR(tabname,15)tabname
FROM SYSCAT.TABLES
WHERE tbspaceid = 5 AND tableid = 17

```

The following is an example of output from this query.

| TABSCHEMA | TABNAME |
|-----------|---------|
| SAMP | NATION |

1 record(s) selected.

SNAP_GET_TBSP_QUIESCER table function

The SNAP_GET_TBSP_QUIESCER table function returns the same information as the SNAPTbsp_QUIESCER administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_TBSP, SNAP_GET_TBSP_PART, SNAP_GET_TBSP_RANGE, SNAP_GET_CONTAINER table functions, the SNAP_GET_TBSP_QUIESCER table function provides information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 309 on page 1423 for a complete list of information that can be returned.

Syntax

```

▶▶ SNAP_GET_TBSP_QUIESCER ( (dbname [ , member ]) )

```

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TBSP QUIESCER table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_TBSP QUIESCER table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Example

Retrieve information about the quiesced table spaces for database member 1 for the currently connected database.

```
SELECT SUBSTR(TBSP_NAME, 1, 10) AS TBSP_NAME, QUIESCER_TS_ID,  
       QUIESCER_OBJ_ID, QUIESCER_AUTH_ID, QUIESCER_AGENT_ID,  
       QUIESCER_STATE, DBPARTITIONNUM  
FROM TABLE( SYSPROC.SNAP_GET_TBSP QUIESCER( ' ', 1)) AS T
```

The following is an example of output from this query.

```

TBSP_NAME  QUIESCER_TS_ID    QUIESCER_OBJ_ID    QUIESCER_AUTH_ID    ...
-----
USERSPACE1                2                5 SWALKTY                ...

1 record(s) selected.

```

Output from this query (continued).

```

... QUIESCER_AGENT_ID    QUIESCER_STATE DBPARTITIONNUM
... -----
...                65983 EXCLUSIVE                1

```

Information returned

Table 335. Information returned by the SNAPTBSP_QUIESCER administrative view and the SNAP_GET_TBSP_QUIESCER table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| QUIESCER_TS_ID | BIGINT | quiescer_ts_id - Quiescer table space identification |
| QUIESCER_OBJ_ID | BIGINT | quiescer_obj_id - Quiescer object identification |
| QUIESCER_AUTH_ID | VARCHAR(128) | quiescer_auth_id - Quiescer user authorization identification |
| QUIESCER_AGENT_ID | BIGINT | quiescer_agent_id - Quiescer agent identification |
| QUIESCER_STATE | VARCHAR(14) | quiescer_state - Quiescer state. This interface returns a text identifier based on defines in sqlutil.h and is one of: <ul style="list-style-type: none"> • EXCLUSIVE • UPDATE • SHARE |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |

SNAPTBSP_RANGE administrative view and SNAP_GET_TBSP_RANGE table function - Retrieve range snapshot information

The SNAPTBSP_RANGE administrative view and the SNAP_GET_TBSP_RANGE table function return information from a range snapshot.

Important: The SNAPTBSP_RANGE administrative view and SNAP_GET_TBSP_RANGE table function are deprecated and have been replaced by “MON_GET_TABLESPACE_RANGE table function - Get information about table space ranges” on page 820.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPTBSP_RANGE administrative view” on page 1378
- “SNAP_GET_TBSP_RANGE table function” on page 1379

SNAPTBSP_RANGE administrative view

This administrative view allows you to retrieve range snapshot information for the currently connected database.

Used with the SNAPTBSP_QUIESCER administrative view and the MON_GET_TABLESPACE and the MON_GET_CONTAINER table functions, the SNAPTBSP_RANGE administrative view provides information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

The schema is SYSIBMADM.

Refer to Table 299 on page 1381 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPTBSP_RANGE administrative view
- CONTROL privilege on the SNAPTBSP_RANGE administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_TBSP_RANGE table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Select information about table space ranges for all database members for the currently connected database.

```

SELECT TBSP_ID, SUBSTR(TBSP_NAME, 1, 15) AS TBSP_NAME, RANGE_NUMBER,
       RANGE_STRIPE_SET_NUMBER, RANGE_OFFSET, RANGE_MAX_PAGE,
       RANGE_MAX_EXTENT, RANGE_START_STRIPE, RANGE_END_STRIPE,
       RANGE_ADJUSTMENT, RANGE_NUM_CONTAINER, RANGE_CONTAINER_ID,
       DBPARTITIONNUM FROM SYSIBMADM.SNAPTBSRANGE
ORDER BY DBPARTITIONNUM

```

The following is an example of output from this query.

| TBSP_ID | TBSP_NAME | RANGE_NUMBER | RANGE_STRIPE_SET_NUMBER | ... |
|---------|--------------|--------------|-------------------------|-----|
| ... | ... | ... | ... | ... |
| 0 | SYSCATSPACE | 0 | 0 | ... |
| 2 | USERSPACE1 | 0 | 0 | ... |
| 3 | SYSTOOLSPACE | 0 | 0 | ... |
| 2 | USERSPACE1 | 0 | 0 | ... |
| 2 | USERSPACE1 | 0 | 0 | ... |

5 record(s) selected.

Output from this query (continued).

| ... | RANGE_OFFSET | RANGE_MAX_PAGE | RANGE_MAX_EXTENT | ... |
|-----|--------------|----------------|------------------|-----|
| ... | ... | ... | ... | ... |
| ... | 0 | 11515 | 2878 | ... |
| ... | 0 | 479 | 14 | ... |
| ... | 0 | 251 | 62 | ... |
| ... | 0 | 479 | 14 | ... |
| ... | 0 | 479 | 14 | ... |

Output from this query (continued).

| ... | RANGE_START_STRIPE | RANGE_END_STRIPE | RANGE_ADJUSTMENT | ... |
|-----|--------------------|------------------|------------------|-----|
| ... | ... | ... | ... | ... |
| ... | 0 | 2878 | 0 | ... |
| ... | 0 | 14 | 0 | ... |
| ... | 0 | 62 | 0 | ... |
| ... | 0 | 14 | 0 | ... |
| ... | 0 | 14 | 0 | ... |

Output from this query (continued).

| ... | RANGE_NUM_CONTAINER | RANGE_CONTAINER_ID | DBPARTITIONNUM |
|-----|---------------------|--------------------|----------------|
| ... | ... | ... | ... |
| ... | 1 | 0 | 0 |
| ... | 1 | 0 | 0 |
| ... | 1 | 0 | 0 |
| ... | 1 | 0 | 1 |
| ... | 1 | 0 | 2 |

SNAP_GET_TBSP_RANGE table function

The SNAP_GET_TBSP_RANGE table function returns the same information as the SNAPTBSRANGE administrative view, but allows you to retrieve the information for a specific database on a specific database member, aggregate of all database members or all database members.

Used with the SNAP_GET_TBSP, SNAP_GET_TBSP_PART, SNAP_GET_TBSP_QUIESCER and SNAP_GET_CONTAINER table functions, the SNAP_GET_TBSP_RANGE table function provides information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 299 on page 1381 for a complete list of information that can be returned.

Syntax

► SNAP_GET_TBSP_RANGE ((*dbname* [, *member*])) ►

The schema is SYSPROC.

Table function parameters

dbname

An input argument of type VARCHAR(128) that specifies a valid database name in the same instance as the currently connected database. Specify a database name that has a directory entry type of either "Indirect" or "Home", as returned by the **LIST DATABASE DIRECTORY** command. Specify NULL or empty string to take the snapshot from the currently connected database.

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If *dbname* is not set to NULL and *member* is set to NULL, -1 is set implicitly for *member*. If this input option is not used, that is, only *dbname* is provided, data is returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If both *dbname* and *member* are set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_TBSP_RANGE table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_TBSP_RANGE table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Select information about the table space range for the table space with `tbody_id = 2` on the currently connected database member.

```

SELECT TBSP_ID, SUBSTR(TBSP_NAME, 1, 15) AS TBSP_NAME, RANGE_NUMBER,
       RANGE_STRIPE_SET_NUMBER, RANGE_OFFSET, RANGE_MAX_PAGE, RANGE_MAX_EXTENT,
       RANGE_START_STRIPE, RANGE_END_STRIPE, RANGE_ADJUSTMENT,
       RANGE_NUM_CONTAINER, RANGE_CONTAINER_ID
FROM TABLE(SNAP_GET_TBSP_RANGE(' ', -1)) AS T WHERE TBSP_ID = 2

```

The following is an example of output from this query.

```

TBSP_ID    TBSP_NAME    RANGE_NUMBER    ...
-----
2 USERSPACE1    0 ...

```

1 record(s) selected.

Output from this query (continued).

```

... RANGE_STRIPE_SET_NUMBER RANGE_OFFSET    RANGE_MAX_PAGE    ...
... -----
... 0 0 3967 ...

```

Output from this query (continued).

```

... RANGE_MAX_EXTENT    RANGE_START_STRIPE    RANGE_END_STRIPE    ...
... -----
... 123 0 123 ...

```

Output from this query (continued).

```

... RANGE_ADJUSTMENT    RANGE_NUM_CONTAINER    RANGE_CONTAINER_ID
... -----
... 0 1 0

```

Information returned

Table 336. Information returned by the *SNAPTbsp_RANGE* administrative view and the *SNAP_GET_TBSP_RANGE* table function

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| RANGE_NUMBER | BIGINT | range_number - Range number |
| RANGE_STRIPE_SET_NUMBER | BIGINT | range_stripe_set_number - Stripe set number |
| RANGE_OFFSET | BIGINT | range_offset - Range offset |
| RANGE_MAX_PAGE | BIGINT | range_max_page_number - Maximum page in range |
| RANGE_MAX_EXTENT | BIGINT | range_max_extent - Maximum extent in range |
| RANGE_START_STRIPE | BIGINT | range_start_stripe - Start stripe |
| RANGE_END_STRIPE | BIGINT | range_end_stripe - End stripe |
| RANGE_ADJUSTMENT | BIGINT | range_adjustment - Range adjustment |
| RANGE_NUM_CONTAINER | BIGINT | range_num_container - Number of containers in range |

Table 336. Information returned by the *SNAPTbsp_Range* administrative view and the *SNAP_GET_TBSP_RANGE* table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| RANGE_CONTAINER_ID | BIGINT | range_container_id - Range container |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

SNAPUTIL administrative view and SNAP_GET_UTIL table function - Retrieve utility_info logical data group snapshot information

The SNAPUTIL administrative view and the SNAP_GET_UTIL table function return snapshot information about the utilities from the utility_info logical data group.

Important: The SNAPUTIL administrative view and the SNAP_GET_UTIL table function are deprecated and have been replaced by the “MON_GET_UTILITY table function - Get utilities running on the database” on page 863.

Depending on if you are using the administrative view or the table function, refer to one of the following sections:

- “SNAPUTIL administrative view” on page 1424
- “SNAP_GET_UTIL table function” on page 1426

SNAPUTIL administrative view

Used in conjunction with the SNAPUTIL_PROGRESS administrative view, the SNAPUTIL administrative view provides the same information as the **LIST UTILITIES SHOW DETAIL** CLP command.

The schema is SYSIBMADM.

Refer to Table 310 on page 1427 for a complete list of information that can be returned.

Authorization

One of the following authorizations is required to use the view:

- SELECT privilege on the SNAPUTIL administrative view
- CONTROL privilege on the SNAPUTIL administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

One of the following is required to use the table function:

- EXECUTE privilege on the SNAP_GET_UTIL table function
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMOINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve a list of utilities and their states on all database members for all active databases in the instance that contains the connected database.

```
SELECT UTILITY_TYPE, UTILITY_PRIORITY, SUBSTR(UTILITY_DESCRIPTION, 1, 72)
      AS UTILITY_DESCRIPTION, SUBSTR(UTILITY_DBNAME, 1, 17) AS
      UTILITY_DBNAME, UTILITY_STATE, UTILITY_INVOKER_TYPE, DBPARTITIONNUM
FROM SYSIBMADM.SNAPUTIL ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| UTILITY_TYPE | UTILITY_PRIORITY | ... |
|--------------|------------------|-----|
| LOAD | - | ... |
| LOAD | - | ... |
| LOAD | - | ... |

3 record(s) selected.

Output from this query (continued).

| ... | UTILITY_DESCRIPTION | ... |
|-----|--|-----|
| ... | ONLINE LOAD DEL AUTOMATIC INDEXING INSERT COPY NO TEST .LOADTEST | ... |
| ... | ONLINE LOAD DEL AUTOMATIC INDEXING INSERT COPY NO TEST .LOADTEST | ... |
| ... | ONLINE LOAD DEL AUTOMATIC INDEXING INSERT COPY NO TEST .LOADTEST | ... |

Output from this query (continued).

| ... | UTILITY_DBNAME | UTILITY_STATE | UTILITY_INVOKER_TYPE | DBPARTITIONNUM |
|-----|----------------|---------------|----------------------|----------------|
| ... | SAMPLE | EXECUTE | USER | 0 |
| ... | SAMPLE | EXECUTE | USER | 1 |
| ... | SAMPLE | EXECUTE | USER | 2 |

SNAP_GET_UTIL table function

The SNAP_GET_UTIL table function returns the same information as the SNAPUTIL administrative view, but allows you to retrieve the information for a specific database member, aggregate of all database members or all database members.

Used in conjunction with the SNAP_GET_UTIL_PROGRESS table function, the SNAP_GET_UTIL table function provides the same information as the **LIST UTILITIES SHOW DETAIL CLP** command.

Refer to Table 310 on page 1427 for a complete list of information that can be returned.

Syntax

► SNAP_GET_UTIL (member) ►

The schema is SYSPROC.

Table function parameter

member

An optional input argument of type INTEGER that specifies a valid database member number. Specify -1 for the current database member, or -2 for an aggregate of all active database members. If this input option is not used, data will be returned from all active database members. An active database member is a member where the database is available for connection and use by applications.

If *member* is set to NULL, an attempt is made to read data from the file created by SNAP_WRITE_FILE procedure. Note that this file could have been created at any time, which means that the data might not be current. If a file with the corresponding snapshot API request type does not exist, then the SNAP_GET_UTIL table function takes a snapshot for the currently connected database and database member number.

Authorization

One of the following authorizations is required:

- EXECUTE privilege on the SNAP_GET_UTIL table function
- DATAACCESS authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMAINT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the function is automatically created.

Examples

Example 1: Retrieve a list of utility ids with their type and state for the currently connected database member on database SAMPLE.

```
SELECT UTILITY_ID, UTILITY_TYPE, UTILITY_STATE
FROM TABLE(SNAP_GET_UTIL(-1)) AS T WHERE UTILITY_DBNAME='SAMPLE'
```

The following is an example of output from this query:

| UTILITY_ID | UTILITY_TYPE | STATE |
|------------|--------------|---------|
| 1 | BACKUP | EXECUTE |

1 record(s) selected.

Example 2: Retrieve a list of utility ids with their type, member number and database partition number for the currently connected database member.

```
SELECT UTILITY_ID, UTILITY_TYPE, MEMBER, DBPARTITIONNUM
FROM TABLE(SNAP_GET_UTIL(-1)) AS T
```

The following is an example of output from this query:

| UTILITY_ID | UTILITY_TYPE | MEMBER | DBPARTITIONNUM |
|------------|--------------|--------|----------------|
| 2 | BACKUP | 2 | 2 |

Information returned

Table 337. Information returned by the SNAPUTIL administrative view and the SNAP_GET_UTIL table function

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|---------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| UTILITY_ID | INTEGER | utility_id - Utility ID. Unique to a database partition. |
| UTILITY_TYPE | VARCHAR(26) | utility_type - Utility type. This interface returns a text identifier based on the defines in sqlmon.h. |
| UTILITY_PRIORITY | INTEGER | utility_priority - Utility priority. Priority if utility supports throttling, otherwise null. |
| UTILITY_DESCRIPTION | VARCHAR(2048) | utility_description - Utility description. Can be null. |
| UTILITY_DBNAME | VARCHAR(128) | utility_dbname - Database operated on by utility |
| UTILITY_START_TIME | TIMESTAMP | utility_start_time - Utility start time |
| UTILITY_STATE | VARCHAR(10) | utility_state - Utility state. This interface returns a text identifier based on the defines in sqlmon.h. |
| UTILITY_INVOKER_TYPE | VARCHAR(10) | utility_invoker_type - Utility invoker type. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> AUTO USER |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| PROGRESS_LIST_ATTR | VARCHAR(10) | progress_list_attr - Current progress list attributes |
| PROGRESS_LIST_CUR_SEQ_NUM | INTEGER | progress_list_current_seq_num - Current progress list sequence number |

Table 337. Information returned by the SNAPUTIL administrative view and the SNAP_GET_UTIL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------|-----------|--|
| MEMBER | SMALLINT | member - Database member monitor element |

SYSINSTALLROUTINES

This procedure has been deprecated.

►—SYSINSTALLROUTINES—(—)—————►

The schema is SYSPROC.

Authorization

One of the following authorities is required to execute the procedure:

- EXECUTE privilege on the procedure
- DATAACCESS authority
- DBADM authority
- SQLADM authority

Default PUBLIC privilege

In a non-restrictive database, EXECUTE privilege is granted to PUBLIC when the procedure is automatically created.

TBSP_UTILIZATION administrative view - Retrieve table space configuration and utilization information

The TBSP_UTILIZATION administrative view returns table space configuration and utilization information.

It retrieves a similar report to the **LIST TABLESPACES** command on a single partitioned database. Its information is based on the SNAPTbsp, SNAPTbsp_PART administrative views and TABLESPACES catalog view.

Important: The TBSP_UTILIZATION administrative view is deprecated and have been replaced by the “MON_TBSP_UTILIZATION - Retrieve monitoring metrics for all table spaces and all database partitions” on page 449.

The schema is SYSIBMADM.

Authorization

One of the following authorizations is required:

- SELECT privilege on the TBSP_UTILIZATION administrative view
- CONTROL privilege on the TBSP_UTILIZATION administrative view
- DATAACCESS authority
- DBADM authority
- SQLADM authority

In addition, to access snapshot monitor data, one of the following authorities is also required:

- SYSMON
- SYSCTRL
- SYSMANT
- SYSADM

Default PUBLIC privilege

In a non-restrictive database, SELECT privilege is granted to PUBLIC when the view is automatically created.

Example

Retrieve the same report as the **LIST TABLESPACES** command on a single partitioned database.

```
SELECT TBSP_ID, SUBSTR(TBSP_NAME,1,20) as TBSP_NAME, TBSP_TYPE,
       TBSP_CONTENT_TYPE, TBSP_STATE FROM SYSIBMADM.TBSP_UTILIZATION
```

The following is an example of output for this query.

```
TBSP_ID    TBSP_NAME          TBSP_TYPE    ...
-----
0 SYSCATSPACE      DMS          ...
1 TEMPSPACE1       DMS          ...
2 USERSPACE1       DMS          ...
3 SYSTOOLSPACE     DMS          ...
4 SYSTOOLSTMPSPACE DMS          ...
```

Output for this query (continued).

```
... TBSP_CONTENT_TYPE TBSP_STATE
... -----
... ANY                NORMAL
... SYSTEMP            NORMAL
... ANY                NORMAL
... ANY                NORMAL
... USRTEMP            NORMAL
```

Information returned

Table 338. Information returned by the TBSP_UTILIZATION administrative view

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TBSP_ID | BIGINT | tablespace_id - Table space identification |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table space name |
| TBSP_TYPE | VARCHAR(10) | tablespace_type - Table space type |
| TBSP_CONTENT_TYPE | VARCHAR(10) | tablespace_content_type - Table space content type |
| TBSP_CREATE_TIME | TIMESTAMP | Creation time of the table space. |
| TBSP_STATE | VARCHAR(256) | tablespace_state - Table space state |

Table 338. Information returned by the TBSP_UTILIZATION administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|--|
| TBSP_TOTAL_SIZE_KB | BIGINT | The total size of the table space in KB, calculated as $\text{total_pages} * \text{pagesize} / 1024$. |
| TBSP_USABLE_SIZE_KB | BIGINT | The total usable size of the table space in KB, calculated as $\text{usable_pages} * \text{pagesize} / 1024$. |
| TBSP_USED_SIZE_KB | BIGINT | The total used size of the table space in KB, calculated as $\text{used_pages} * \text{pagesize} / 1024$. |
| TBSP_FREE_SIZE_KB | BIGINT | The total available size of the table space in KB, calculated as $\text{free_pages} * \text{pagesize} / 1024$. |
| TBSP_UTILIZATION_PERCENT | BIGINT | The utilization of the table space as a percentage. Calculated as $(\text{used_pages} / \text{usable_pages}) * 100$, if usable_pages is available. Otherwise, -1 will be displayed. |
| TBSP_TOTAL_PAGES | BIGINT | tablespace_total_pages - Total pages in table space |
| TBSP_USABLE_PAGES | BIGINT | tablespace_usable_pages - Usable pages in table space |
| TBSP_USED_PAGES | BIGINT | tablespace_used_pages - Used pages in table space |
| TBSP_FREE_PAGES | BIGINT | tablespace_free_pages - Free pages in table space |
| TBSP_PAGE_TOP | BIGINT | tablespace_page_top - Table space high water mark |
| TBSP_PAGE_SIZE | INTEGER | tablespace_page_size - Table space page size |
| TBSP_EXTENT_SIZE | INTEGER | tablespace_extent_size - Table space extent size |
| TBSP_PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table space prefetch size |
| TBSP_MAX_SIZE | BIGINT | tablespace_max_size - Maximum table space size |
| TBSP_INCREASE_SIZE | BIGINT | tablespace_increase_size - Increase size in bytes |
| TBSP_INCREASE_SIZE_PERCENT | SMALLINT | tablespace_increase_size_percent - Increase size by percent |
| TBSP_LAST_RESIZE_TIME | TIMESTAMP | tablespace_last_resize_time - Time of last successful resize |
| TBSP_LAST_RESIZE_FAILED | SMALLINT | tablespace_last_resize_failed - Last resize attempt failed |
| TBSP_USING_AUTO_STORAGE | SMALLINT | tablespace_using_auto_storage - Table space enabled for automatic storage |

Table 338. Information returned by the TBSP_UTILIZATION administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|--------------|---|
| TBSP_AUTO_RESIZE_ENABLED | SMALLINT | tablespace_auto_resize_enabled - Table space automatic resizing enabled |
| DBPGNAME | VARCHAR(128) | Name of the database partition group for the table space. |
| TBSP_NUM_CONTAINERS | BIGINT | tablespace_num_containers - Number of containers in table space |
| REMARKS | VARCHAR(254) | User-provided comment. |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |

WLM_GET_ACTIVITY_DETAILS - Return detailed information about a specific activity

This function returns detailed information about a specific activity identified by its application handle, unit of work ID, and activity ID. This information includes details about any thresholds that the activity has violated.

Note: This table function has been deprecated and replaced by “MON_GET_ACTIVITY_DETAILS table function - Return information about an activity as an XML document” on page 533.

This function returns basic statistics of one or more service subclasses.

Syntax

```

►►WLM_GET_ACTIVITY_DETAILS(—application_handle—,—uow_id—,——————►
►activity_id—,—member—)—————►

```

The schema is SYSPROC.

Table function parameters

application_handle

An input argument of type BIGINT that specifies a valid application handle. If the argument is null, no rows are returned from this function. If the argument is null, an SQL171N error is returned.

uow_id

An input argument of type INTEGER that specifies a valid unit of work identifier unique within the application. If the argument is null, no rows are returned from this function. If the argument is null, an SQL171N error is returned.

activity_id

An input argument of type INTEGER that specifies a valid activity ID unique within the unit of work. If the argument is null, no rows are returned from this function. If the argument is null, an SQL171N error is returned.

member

An input argument of type INTEGER that specifies a valid member number in the same instance as the currently connected database when calling this function. Specify -1 for the current database member, or -2 for all database members. If a null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Example

Detailed information about an individual activity can be obtained by using the WLM_GET_ACTIVITY_DETAILS table function. This table function returns activity information as name-value pairs for each member. This example is restricted to showing only an eleven member subset of the name-value pairs for each member for an activity identified by an application handle of 1, a unit of work ID of 1 and an activity ID of 5. For a complete list of name-value pairs, see Table 340 on page 1577 and Table 341 on page 1579.

```
SELECT SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(NAME, 1, 20) AS NAME,
       SUBSTR(VALUE, 1, 30) AS VALUE
FROM TABLE(WLM_GET_ACTIVITY_DETAILS(1, 1, 5, -2)) AS ACTDETAIL
WHERE NAME IN ('APPLICATION_HANDLE',
              'COORD_PARTITION_NUM',
              'LOCAL_START_TIME',
              'UOW_ID',
              'ACTIVITY_ID',
              'PARENT_UOW_ID',
              'PARENT_ACTIVITY_ID',
              'ACTIVITY_TYPE',
              'NESTING_LEVEL',
              'INVOCATION_ID',
              'ROUTINE_ID')
ORDER BY PART
```

The following is an example of output from this query.

| PART | NAME | VALUE |
|------|---------------------|----------------------------|
| 0 | APPLICATION_HANDLE | 1 |
| 0 | COORD_PARTITION_NUM | 0 |
| 0 | LOCAL_START_TIME | 2005-11-25-18.52.49.343000 |
| 0 | UOW_ID | 1 |
| 0 | ACTIVITY_ID | 5 |
| 0 | PARENT_UOW_ID | 1 |
| 0 | PARENT_ACTIVITY_ID | 3 |
| 0 | ACTIVITY_TYPE | READ_DML |
| 0 | NESTING_LEVEL | 0 |
| 0 | INVOCATION_ID | 1 |
| 0 | ROUTINE_ID | 0 |

```

1  APPLICATION_HANDLE  1
1  COORD_PARTITION_NUM 0
1  LOCAL_START_TIME   2005-11-25-18.52.49.598000
1  UOW_ID              1
1  ACTIVITY_ID         5
1  PARENT_UOW_ID
1  PARENT_ACTIVITY_ID
1  ACTIVITY_TYPE       READ_DML
1  NESTING_LEVEL       0
1  INVOCATION_ID       1
1  ROUTINE_ID          0

```

Usage note

An ACTIVITY_STATE of QUEUED means that the coordinator activity has made a RPC to the catalog member to obtain threshold tickets and has not yet received a response. Seeing this state might indicate that the activity has been queued by WLM or, over short periods of time, might just indicate that the activity is in the process of obtaining its tickets. To obtain a more accurate picture of whether or not the activity is really being queued, one can determine which agent is working on the activity (using the WLM_GET_SERVICE_CLASS_AGENTS table function) and find out whether this agent's event_object at the catalog member has a value of WLM_QUEUE.

Information returned

Table 339. Information returned for WLM_GET_ACTIVITY_DETAILS

| Column Name | Data Type | Description |
|----------------|---------------|---|
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| NAME | VARCHAR(256) | Element name. See Table 340 and Table 341 on page 1579 for possible values. |
| VALUE | VARCHAR(1024) | Element values. See Table 340 and Table 341 on page 1579 for possible values. |

Table 340. Elements returned

| Element Name | Description |
|-----------------------------|---|
| ACTIVITY_ID | activity_id - Activity ID monitor element |
| ACTIVITY_STATE | activity_state - Activity state monitor element |
| ACTIVITY_TYPE | activity_type - Activity type monitor element |
| APPLICATION_HANDLE | application_handle - Application handle monitor element |
| COORD_PARTITION_NUM | coord_partition_num - Coordinator partition number monitor element |
| DATABASE_WORK_ACTION_SET_ID | If this activity has been mapped to a work action set that has been applied to the database, this column contains the ID of the work action set. This column contains 0 if the activity has not been mapped to a work action set that has been applied to the database. |
| DATABASE_WORK_CLASS_ID | If this activity has been mapped to a work action set that has been applied to the database, this column contains the ID of the work class of this activity. This column contains 0 if the activity has not been mapped to a work action set that has been applied to the database. |
| EFFECTIVE_ISOLATION | effective_isolation - Effective isolation monitor element |

Table 340. Elements returned (continued)

| Element Name | Description |
|----------------------------------|---|
| EFFECTIVE_LOCK_TIMEOUT | effective_lock_timeout - Effective lock timeout monitor element |
| EFFECTIVE_QUERY_DEGREE | effective_query_degree - Effective query degree monitor element |
| ENTRY_TIME | entry_time - Entry time monitor element |
| INVOCATION_ID | invocation_id - Invocation ID monitor element |
| LAST_REFERENCE_TIME | last_reference_time - Last reference time monitor element |
| LOCAL_START_TIME | local_start_time - Local start time monitor element |
| NESTING_LEVEL | nesting_level - Nesting level monitor element |
| PACKAGE_NAME | package_name - Package name monitor element |
| PACKAGE_SCHEMA | package_schema - Package schema monitor element |
| PACKAGE_VERSION_ID | package_version_id - Package version monitor element |
| PARENT_ACTIVITY_ID | parent_activity_id - Parent activity ID monitor element |
| PARENT_UOW_ID | Unique unit of work identifier within an application. Refers to the original unit of work this activity's parent activity started in. Returns an empty string if the activity has no parent activity or when at a remote member. |
| QUERY_COST_ESTIMATE | query_cost_estimate - Query cost estimate monitor element |
| ROUTINE_ID | routine_id - Routine ID monitor element |
| ROWS_FETCHED | rows_fetched - Rows fetched monitor element |
| ROWS_MODIFIED | rows_modified - Rows modified monitor element |
| SECTION_NUMBER | section_number - Section number monitor element |
| SERVICE_CLASS_ID | service_class_id - Service class ID monitor element |
| SERVICE_CLASS_WORK_ACTION_SET_ID | If this activity has been mapped to a work action set that has been applied to a service class, this column contains the ID of the work action set. This column contains 0 if the activity has not been mapped to a work action set that has been applied to a service class. |
| SERVICE_CLASS_WORK_CLASS_ID | If this activity has been mapped to a work action set that has been applied to a service class, this column contains the ID of the work class of this activity. This column contains 0 if the activity has not been mapped to a work action set that has been applied to a service class. |
| STMT_PKG_CACHE_ID | stmt_pkgcache_id - Statement package cache identifier monitor element |
| STMT_TEXT | stmt_text - SQL statement text monitor element |
| SYSTEM_CPU_TIME | system_cpu_time - System CPU time monitor element |
| UOW_ID | uow_id - Unit of work ID monitor element |
| USER_CPU_TIME | user_cpu_time - User CPU time monitor element |
| UTILITY_ID | utility_id - Utility ID monitor element |

Important: The WLM_GET_ACTIVITY_DETAILS table function shows only the thresholds that are currently being applied to an activity.

The following elements are returned only if the corresponding thresholds apply to the activity.

Table 341. Elements returned if applicable

| Element Name | Description |
|---|---|
| ACTIVITYTOTALTIME_THRESHOLD_ID | activitytotaltime_threshold_id - Activity total time threshold ID monitor element |
| ACTIVITYTOTALTIME_THRESHOLD_VALUE | activitytotaltime_threshold_value - Activity total time threshold value monitor element |
| ACTIVITYTOTALTIME_THRESHOLD_VIOLATED | activitytotaltime_threshold_violated - Activity total time threshold violated monitor element |
| CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_ID | concurrentdbcoordactivities_db_threshold_id - Concurrent database coordinator activities database threshold ID monitor element |
| CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_QUEUED | concurrentdbcoordactivities_db_threshold_queued - Concurrent database coordinator activities database threshold queued monitor element |
| CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_VALUE | concurrentdbcoordactivities_db_threshold_value - Concurrent database coordinator activities database threshold value monitor element |
| CONCURRENTDBCOORDACTIVITIES_DB_THRESHOLD_VIOLATED | concurrentdbcoordactivities_db_threshold_violated - Concurrent database coordinator activities database threshold violated monitor element |
| CONCURRENTDBCOORDACTIVITIES_SUBCLASS_THRESHOLD_ID | concurrentdbcoordactivities_subclass_threshold_id - Concurrent database coordinator activities service subclass threshold ID monitor element |
| CONCURRENTDBCOORDACTIVITIES_SUBCLASS_THRESHOLD_QUEUED | 'Yes' indicates that the activity was queued by the CONCURRENTDBCOORDACTIVITIES_SUBCLASS threshold. 'No' indicates that the activity was not queued. |
| CONCURRENTDBCOORDACTIVITIES_SUBCLASS_THRESHOLD_VALUE | concurrentdbcoordactivities_subclass_threshold_value - Concurrent database coordinator activities service subclass threshold value monitor element |
| CONCURRENTDBCOORDACTIVITIES_SUBCLASS_THRESHOLD_VIOLATED | 'Yes' indicates that the activity violated the CONCURRENTDBCOORDACTIVITIES_SUBCLASS threshold. 'No' indicates that the activity has not yet violated the threshold. |
| CONCURRENTDBCOORDACTIVITIES_SUPERCLASS_THRESHOLD_ID | The ID of the CONCURRENTDBCOORDACTIVITIES_SUPERCLASS threshold that was applied to the activity. |
| CONCURRENTDBCOORDACTIVITIES_SUPERCLASS_THRESHOLD_QUEUED | 'Yes' indicates that the activity was queued by the CONCURRENTDBCOORDACTIVITIES_SUPERCLASS threshold. 'No' indicates that the activity was not queued. |
| CONCURRENTDBCOORDACTIVITIES_SUPERCLASS_THRESHOLD_VALUE | The upper bound of the CONCURRENTDBCOORDACTIVITIES_SUPERCLASS threshold that was applied to the activity. |
| CONCURRENTDBCOORDACTIVITIES_SUPERCLASS_THRESHOLD_VIOLATED | 'Yes' indicates that the activity violated the CONCURRENTDBCOORDACTIVITIES_SUPERCLASS threshold. 'No' indicates that the activity has not yet violated the threshold. |

Table 341. Elements returned if applicable (continued)

| Element Name | Description |
|--|--|
| CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET_THRESHOLD_ID | The ID of the CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET threshold that was applied to the activity. |
| CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET_THRESHOLD_QUEUED | 'Yes' indicates that the activity was queued by the CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET threshold. 'No' indicates that the activity was not queued. |
| CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET_THRESHOLD_VALUE | The upper bound of the CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET threshold that was applied to the activity. |
| CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET_THRESHOLD_VIOLATED | 'Yes' indicates that the activity violated the CONCURRENTDBCOORDACTIVITIES_WORK_ACTION_SET threshold. 'No' indicates that the activity has not yet violated the threshold. |
| CONCURRENTWORKLOADACTIVITIES_THRESHOLD_ID | The ID of the CONCURRENTWORKLOADACTIVITIES threshold that was applied to the activity. |
| CONCURRENTWORKLOADACTIVITIES_THRESHOLD_VALUE | The upper bound of the CONCURRENTWORKLOADACTIVITIES threshold that was applied to the activity. |
| CONCURRENTWORKLOADACTIVITIES_THRESHOLD_VIOLATED | 'Yes' indicates that the activity violated the CONCURRENTWORKLOADACTIVITIES threshold. 'No' indicates that the activity has not yet violated the threshold. |
| ESTIMATEDSQLCOST_THRESHOLD_ID | estimatedsqlcost_threshold_id - Estimated SQL cost threshold ID monitor element |
| ESTIMATEDSQLCOST_THRESHOLD_VALUE | estimatedsqlcost_threshold_value - Estimated SQL cost threshold value monitor element |
| ESTIMATEDSQLCOST_THRESHOLD_VIOLATED | estimatedsqlcost_threshold_violated - Estimated SQL cost threshold violated monitor element |
| SQLROWSRETURNED_THRESHOLD_ID | sqlrowsreturned_threshold_id - SQL rows read returned threshold ID monitor element |
| SQLROWSRETURNED_THRESHOLD_VALUE | sqlrowsreturned_threshold_value - SQL rows read returned threshold value monitor element |
| SQLROWSRETURNED_THRESHOLD_VIOLATED | sqlrowsreturned_threshold_violated - SQL rows read returned threshold violated monitor element |
| SQLTEMPSPACE_THRESHOLD_ID | sqltempespace_threshold_id - SQL temporary space threshold ID monitor element |
| SQLTEMPSPACE_THRESHOLD_VALUE | sqltempespace_threshold_value - SQL temporary space threshold value monitor element |
| SQLTEMPSPACE_THRESHOLD_VIOLATED | sqltempespace_threshold_violated - SQL temporary space threshold violated monitor element |

WLM_GET_SERVICE_CLASS_AGENTS _V97 - List agents running in a service class

The WLM_GET_SERVICE_CLASS_AGENTS_V97 table function returns the list of agents, fenced mode processes (db2fmp processes), and system entities on a

specified member that are running in a specified service class or on behalf of a specified application. The system entities are non-agent threads and processes, such as page cleaners and prefetchers.

Note: This table function has been deprecated and replaced by the WLM_GET_SERVICE_CLASS_AGENTS table function.

Syntax

```
►►WLM_GET_SERVICE_CLASS_AGENTS_V97—(—service_superclass_name—,—————►
►service_subclass_name—,—application_handle—,—member—)—————►◄
```

The schema is SYSPROC.

Table function parameters

service_superclass_name

An input argument of type VARCHAR(128) that specifies the name of a service superclass in the currently connected database. If the argument is null or an empty string, data is retrieved for all the superclasses in the database.

service_subclass_name

An input argument of type VARCHAR(128) that refers to a specific subclass within a superclass. If the argument is null or an empty string, data is retrieved for all the subclasses in the database.

application_handle

An input argument of type BIGINT that specifies the application handle for which agent information is to be returned. If the argument is null, data is retrieved for all applications in the database. An application handle of 0 returns the system entities only.

member

An input argument of type INTEGER that specifies the member number in the same instance as the currently connected database. Specify -1 for the current database member, or -2 for all database members. If a null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Examples

Example 1

The following query returns a list of agents that are associated with application handle 1 for all database members. You can determine the application handle by using the **LIST APPLICATIONS** command or the **WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97** table function.

```
SELECT SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHANDLE,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(CHAR(AGENT_TID),1,9) AS AGENT_TID,
       SUBSTR(AGENT_TYPE,1,11) AS AGENTTYPE,
       SUBSTR(AGENT_STATE,1,10) AS AGENTSTATE,
       SUBSTR(REQUEST_TYPE,1,12) AS REQTYPE,
       SUBSTR(CHAR(UOW_ID),1,6) AS UOW_ID,
       SUBSTR(CHAR(ACTIVITY_ID),1,6) AS ACT_ID
FROM TABLE(WLM_GET_SERVICE_CLASS_AGENTS_V97(CAST(NULL AS VARCHAR(128)),
      CAST(NULL AS VARCHAR(128)), 1, -2)) AS SCDETAILS
ORDER BY APPHANDLE, PART, AGENT_TID
```

Sample output is as follows:

| APPHANDLE | PART | AGENT_TID | AGENTTYPE | AGENTSTATE | REQTYPE | UOW_ID | ACT_ID |
|-----------|------|-----------|-------------|------------|--------------|--------|--------|
| 1 | 0 | 3 | COORDINATOR | ACTIVE | FETCH | 1 | 5 |
| 1 | 0 | 4 | SUBAGENT | ACTIVE | SUBSECTION:1 | 1 | 5 |
| 1 | 1 | 2 | SUBAGENT | ACTIVE | SUBSECTION:2 | 1 | 5 |

The output shows a coordinator agent and a subagent on member 0 and a subagent on member 1 operating on behalf of an activity with UOW ID 1 and activity ID 5. The AGENTTYPE column with a value of COORDINATOR has a value of FETCH for the REQTYPE column (which indicates the main or initial request type). This means that the type of request is a fetch request for the coordinator agent.

Example 2

The following query determines which lock an agent is waiting on:

```
select event_object, event_type, event_state, varchar
      (event_object_name, 30)
as event_object_name
from table
      (wlm_get_service_class_agents_v97('',' ',cast(NULL as bigint), -1)) as t
```

Sample output is as follows:

| EVENT_OBJECT | EVENT_TYPE | EVENT_STATE |
|--------------|------------|-------------|
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | WAIT | IDLE |
| LOCK | ACQUIRE | IDLE |
| ROUTINE | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |

```
EVENT_OBJECT_NAME
-----
-
-
-
-
-
-
-
-
-
-
0200050000000000000000000054
-
-
-
-
-
-
-
-
-
-
```

21 record(s) selected.

Using the same query at a later time shows that the WLM threshold has queued an agent:

| EVENT_OBJECT | EVENT_TYPE | EVENT_STATE |
|--------------|------------|-------------|
| ----- | | |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| WLM_QUEUE | WAIT | IDLE |
| ROUTINE | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |
| REQUEST | PROCESS | EXECUTING |

```
EVENT_OBJECT_NAME
-----
-
-
-
-
-
-
-
-
```

21 record(s) selected.

The parameters are, in effect, ANDed together. That is, if you specify conflicting input parameters, such as a service superclass SUP_A and a subclass SUB_B such that SUB_B is not a subclass of SUP_A, no rows are returned.

Table 342. Information returned by WLM_GET_SERVICE_CLASS_AGENTS_V97

1584 Db2 11.1 for Linux, UNIX, and Windows: Built-In Routines and Views

Table 342. Information returned by WLM_GET_SERVICE_CLASS_AGENTS_V97 (continued)

| Column name | Data type | Description |
|-----------------|--------------|---|
| AGENT_TYPE | VARCHAR (32) | Agent type. The agent types are as follows: <ul style="list-style-type: none"> • COORDINATOR • OTHER • PDBSUBAGENT • SMPSUBAGENT If the value is COORDINATOR, the agent ID might change in concentrator environments. |
| SMP_COORDINATOR | INTEGER | Indication of whether the agent is an SMP coordinator: 1 for yes and 0 for no. |
| AGENT_SUBTYPE | VARCHAR (32) | Agent subtype. The possible subtypes are as follows: <ul style="list-style-type: none"> • DSS • OTHER • RPC • SMP |
| AGENT_STATE | VARCHAR (32) | Indication of whether an agent is associated or active. The possible values are: <ul style="list-style-type: none"> • ASSOCIATED • ACTIVE |
| EVENT_TYPE | VARCHAR (32) | Type of event last processed by this agent. The possible values are as follows: <ul style="list-style-type: none"> • ACQUIRE • PROCESS • WAIT See Table 111 on page 557 for more information about possible values for this column. |
| EVENT_OBJECT | VARCHAR (32) | Object of the event last processed by this agent. The possible values are as follows: <ul style="list-style-type: none"> • COMPRESSION_DICTIONARY_BUILD • IMPLICIT_REBIND • INDEX_RECREATE • LOCK • LOCK_ESCALATION • QP_QUEUE • REMOTE_REQUEST • REQUEST • ROUTINE • WLM_QUEUE See Table 111 on page 557 for more information about possible values for this column. |
| EVENT_STATE | VARCHAR (32) | State of the event last processed by this agent. The possible values are as follows: <ul style="list-style-type: none"> • EXECUTING • IDLE See Table 111 on page 557 for more information about possible values for this column. |

Table 342. Information returned by `WLM_GET_SERVICE_CLASS_AGENTS_V97` (continued)

| Column name | Data type | Description |
|--------------|--------------|--|
| REQUEST_ID | VARCHAR (64) | Request ID. This value is unique only in combination with the value of <i>application_handle</i> . You can use this combination to distinguish between one request that is taking a long time and multiple requests; for example, to distinguish between one long fetch and multiple fetches. |
| REQUEST_TYPE | VARCHAR (32) | Type of request. The possible values are as follows: <ul style="list-style-type: none"> • For coordinator agents: <ul style="list-style-type: none"> – CLOSE – COMMIT – COMPILE – DESCRIBE – EXCSQLSET – EXECIMMD – EXECUTE – FETCH – INTERNAL <i>number</i>, where <i>number</i> is the value of the internal constant – OPEN – PREPARE – REBIND – REDISTRIBUTE – REORG – ROLLBACK – RUNSTATS • For subagents with an AGENT_SUBTYPE of DSS or SMP: <ul style="list-style-type: none"> – If the subsection number is nonzero, the subsection number in the form SUBSECTION:<i>subsection number</i>; otherwise, returns NULL. |

Table 342. Information returned by WLM_GET_SERVICE_CLASS_AGENTS_V97 (continued)

| Column name | Data type | Description |
|--------------------------|----------------|---|
| REQUEST_TYPE (continued) | VARCHAR (32) | <ul style="list-style-type: none"> For subagents with an AGENT_SUBTYPE of RPC: <ul style="list-style-type: none"> ABP CATALOG INTERNAL REORG RUNSTATS WLM For subagents with a SUBTYPE of OTHER: <ul style="list-style-type: none"> ABP APP_RBSVPT APP_RELSVPT BACKUP CLOSE EXTERNAL_RBSVPT EVMON FORCE FORCE_ALL INTERNAL <i>number</i>, where <i>number</i> is the value of the internal constant INTERRUPT NOOP (if there is no request) QP REDISTRIBUTE STMT_RBSVPT STOP_USING UPDATE_DBM_CFG WLM |
| NESTING_LEVEL | INTEGER | nesting_level - Nesting level monitor element |
| INVOCATION_ID | INTEGER | invocation_id - Invocation ID monitor element |
| ROUTINE_ID | INTEGER | routine_id - Routine ID monitor element |
| EVENT_OBJECT_NAME | VARCHAR (1024) | Event object name. If the value of EVENT_OBJECT is LOCK, the value of this column is the name of the lock that the agent is waiting on. If the value of EVENT_OBJECT is WLM_QUEUE, the value of the column is the name of the WLM threshold that the agent is queued on. Otherwise, the value is NULL. |
| APPLICATION_NAME | VARCHAR (128) | appl_name - Application name |
| APPLICATION_ID | VARCHAR (128) | appl_id - Application ID |
| CLIENT_PID | BIGINT | client_pid - Client process ID |
| SESSION_AUTH_ID | VARCHAR (128) | session_auth_id - Session authorization ID |
| REQUEST_START_TIME | TIMESTAMP | Time that the agent started processing the request on which it is currently working |

Table 342. Information returned by WLM_GET_SERVICE_CLASS_AGENTS_V97 (continued)

| Column name | Data type | Description |
|------------------------------|------------------------------|---|
| AGENT_STATE_LAST_UPDATE_TIME | TIMESTAMP | The last time that the event, being processed by the agent, was changed. The event currently processed by the agent is identified by the EVENT_TYPE, EVENT_OBJECT, and EVENT_STATE columns. |
| EXECUTABLE_ID | VARCHAR (32)
FOR BIT DATA | executable_id - Executable ID monitor element |

Note: The possible combinations of EVENT_STATE, EVENT_TYPE, EVENT_OBJECT and EVENT_OBJECT_NAME column values are listed in the following table.

Table 343. Possible combinations for EVENT_STATE, EVENT_TYPE, EVENT_OBJECT and EVENT_OBJECT_NAME column values

| Event description | EVENT_STATE value | EVENT_TYPE value | EVENT_OBJECT value | EVENT_OBJECT_NAME value |
|--|-------------------|------------------|--------------------|-------------------------|
| Acquire lock | IDLE | ACQUIRE | LOCK | Lock name |
| Escalate lock | EXECUTING | PROCESS | LOCK_ESCALATION | NULL |
| Process request | EXECUTING | PROCESS | REQUEST | NULL |
| Wait for a new request | IDLE | WAIT | REQUEST | NULL |
| Wait for a request to be processed at a remote partition | IDLE | WAIT | REMOTE_REQUEST | NULL |
| Wait on a WLM threshold queue | IDLE | WAIT | WLM_QUEUE | Threshold name |
| Process a routine | EXECUTING | PROCESS | ROUTINE | NULL |
| Re-create an index | EXECUTING | PROCESS | INDEX_RECREATE | NULL |
| Build compression dictionary | EXECUTING | PROCESS | COMP_DICT_BLD | NULL |
| Implicit rebind | EXECUTING | PROCESS | IMPLICIT_REBIND | NULL |

WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97 - List of workload occurrences

The WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97 function returns the list of all workload occurrences running in a specified service class on a particular member. A workload occurrence is a specific database connection whose attributes match the definition of a workload and hence is associated with or assigned to the workload.

Note: This table function has been deprecated and replaced by the WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES table function.

A workload occurrence is a specific database connection whose attributes match with the definition of a workload and hence is associated with or assigned to the workload.

Syntax

```
►►—WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97—(—service_superclass_name—,—►►
►—service_subclass_name—,—member—)—►►
```

The schema is SYSPROC.

Table function parameters

service_superclass_name

An input argument of type VARCHAR(128) that specifies the name of a service superclass in the currently connected database. If the argument is null or an empty string, the data is retrieved for all the superclasses in the database that match the values of the other parameters.

service_subclass_name

Target service subclass for the workload occurrence. Any work submitted by this workload occurrence will run in this service subclass under the target service superclass with the exception of activities that are mapped, or remapped, to a different subclass.

member

An input argument of type INTEGER that specifies the number of a member in the same instance as the currently connected database. Specify -1 for the current database member, or -2 for all database members. If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Example

If an administrator wants to see what workload occurrences are running on the system as a whole, the administrator can call the WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97 function by specifying a null value or an empty string for *service_superclass_name* and *service_subclass_name* and -2 for *member*:

```
SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(CHAR(COORD_PARTITION_NUM),1,4) AS COORDPART,
```

```

SUBSTR(CHAR(APPLICATION_HANDLE),1,7) AS APPHNDL,
SUBSTR(WORKLOAD_NAME,1,22) AS WORKLOAD_NAME,
SUBSTR(CHAR(WORKLOAD_OCCURRENCE_ID),1,6) AS WLO_ID
FROM TABLE(WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97
(CAST(NULL AS VARCHAR(128)), CAST(NULL AS VARCHAR(128)), -2))
AS SCINFO
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, PART, APPHNDL,
WORKLOAD_NAME, WLO_ID

```

If the system has four database members and is currently running two workloads, the previous query produces results such as the following ones:

```

SUPERCLASS_NAME  SUBCLASS_NAME  PART  COORDPART  ...
-----
SYSDEFAULTMAINTENAN  SYSDEFAULTSUBCLASS  0      0      ...
SYSDEFAULTSYSTEMCLA  SYSDEFAULTSUBCLASS  0      0      ...
SYSDEFAULTUSERCLASS  SYSDEFAULTSUBCLASS  0      0      ...
SYSDEFAULTUSERCLASS  SYSDEFAULTSUBCLASS  0      0      ...
SYSDEFAULTUSERCLASS  SYSDEFAULTSUBCLASS  1      0      ...
SYSDEFAULTUSERCLASS  SYSDEFAULTSUBCLASS  1      0      ...
SYSDEFAULTUSERCLASS  SYSDEFAULTSUBCLASS  2      0      ...
SYSDEFAULTUSERCLASS  SYSDEFAULTSUBCLASS  2      0      ...
SYSDEFAULTUSERCLASS  SYSDEFAULTSUBCLASS  3      0      ...
SYSDEFAULTUSERCLASS  SYSDEFAULTSUBCLASS  3      0      ...
... APPHNDL  WORKLOAD_NAME  WLO_ID
... -----
... -      -      -
... -      -      -
... 1      SYSDEFAULTUSERWORKLOAD 1
... 2      SYSDEFAULTUSERWORKLOAD 2
... 1      SYSDEFAULTUSERWORKLOAD 1
... 2      SYSDEFAULTUSERWORKLOAD 2
... 1      SYSDEFAULTUSERWORKLOAD 1
... 2      SYSDEFAULTUSERWORKLOAD 2
... 1      SYSDEFAULTUSERWORKLOAD 1
... 2      SYSDEFAULTUSERWORKLOAD 2

```

Usage note

The parameters are, in effect, ANDed together. That is, if you specify conflicting input parameters, such as a service superclass SUP_A and a subclass SUB_B such that SUB_B is not a subclass of SUP_A, no rows are returned.

Note: Statistics reported for the workload occurrence (for example, coord_act_completed_total) are reset at the beginning of each unit of work when they are combined with the corresponding workload statistics.

Information returned

Table 344. Information returned for WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97

| Column name | Data type | Description |
|-------------------------|--------------|--|
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name monitor element |
| SERVICE_SUBCLASS_NAME | VARCHAR(128) | service_subclass_name - Service subclass name monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| COORD_PARTITION_NUM | SMALLINT | coord_partition_num - Coordinator partition number monitor element |

Table 344. Information returned for WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97 (continued)

| Column name | Data type | Description |
|---------------------------|--------------|--|
| APPLICATION_HANDLE | BIGINT | application_handle - Application handle monitor element |
| WORKLOAD_NAME | VARCHAR(128) | workload_name - Workload name monitor element |
| WORKLOAD_OCCURRENCE_ID | INTEGER | workload_occurrence_id - Workload occurrence identifier monitor element |
| UOW_ID | INTEGER | uow_id - Unit of work ID monitor element |
| WORKLOAD_OCCURRENCE_STATE | VARCHAR(32) | workload_occurrence_state - Workload occurrence state monitor element |
| SYSTEM_AUTH_ID | VARCHAR(128) | system_auth_id - System authorization identifier monitor element |
| SESSION_AUTH_ID | VARCHAR(128) | session_auth_id - Session authorization ID monitor element |
| APPLICATION_NAME | VARCHAR(128) | appl_name - Application name monitor element |
| CLIENT_WRKSTNNAME | VARCHAR(255) | client_wrkstnname - Client workstation name monitor element |
| CLIENT_ACCTNG | VARCHAR(255) | client_acctng - Client accounting string monitor element |
| CLIENT_USER | VARCHAR(255) | Current value of the CLIENT_USERID special register for this workload occurrence. |
| CLIENT_APPLNAME | VARCHAR(255) | client_applname - Client application name monitor element |
| COORD_ACT_COMPLETED_TOTAL | INTEGER | coord_act_completed_total - Coordinator activities completed total monitor element |
| COORD_ACT_ABORTED_TOTAL | INTEGER | coord_act_aborted_total - Coordinator activities aborted total monitor element |
| COORD_ACT_REJECTED_TOTAL | INTEGER | coord_act_rejected_total - Coordinator activities rejected total monitor element |
| CONCURRENT_ACT_TOP | INTEGER | concurrent_act_top - Concurrent activity top monitor element |
| ADDRESS | VARCHAR(255) | address - IP address from which the connection was initiated |

WLM_GET_SERVICE_SUBCLASS_STATS_V97 - return statistics of service subclasses

The WLM_GET_SERVICE_SUBCLASS_STATS_V97 table function returns basic statistics of one or more service subclasses.

Note: This table function has been deprecated and replaced by the WLM_GET_SERVICE_SUBCLASS_STATS table function.

Refer to Table 345 on page 1594 for a complete list of information that can be returned.

Syntax

```

▶▶—WLM_GET_SERVICE_SUBCLASS_STATS_V97—(—service_superclass_name—,—————▶
▶—service_subclass_name—,—member—)—————▶▶

```

The schema is SYSPROC.

Table function parameters

service_superclass_name

An input argument of type VARCHAR(128) that specifies the name of a service superclass in the currently connected database. If the argument is null or an empty string, the data is retrieved for all of the superclasses in the database.

service_subclass_name

An input argument of type VARCHAR(128) that specifies the name of a service subclass in the currently connected database. If the argument is null or an empty string, the data is retrieved for all of the subclasses in the database.

member

An input argument of type INTEGER that specifies a valid member number in the same instance as the currently connected database. Specify -1 for the current database member, or -2 for all database members. If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Examples

Example 1: Because every activity must be mapped to a database service class before being run, you can monitor the global state of the system by using the service class statistics table functions and querying all of the service classes on all members. In the following example, a null value is passed for *service_superclass_name* and *service_subclass_name* to return statistics for all service classes, and the value -2 is specified for *dbpartitionnum* to return statistics for all partitions:

```

SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       CAST(COORD_ACT_LIFETIME_AVG / 1000 AS DECIMAL(9,3))
       AS AVGLIFETIME,
       CAST(COORD_ACT_LIFETIME_STDDEV / 1000 AS DECIMAL(9,3))
       AS STDDEV_LIFETIME,
       SUBSTR(CAST(LAST_RESET AS VARCHAR(30)),1,16) AS LAST_RESET
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS_V97(CAST(NULL AS VARCHAR(128)),
        CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, PART

```

The statement returns service class statistics such as average activity lifetime and standard deviation in seconds, as shown in the following sample output:

| SUPERCLASS_NAME | SUBCLASS_NAME | PART | ... |
|---------------------|--------------------|----------------|------------------|
| ----- | ----- | ---- | --- |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 0 | ... |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 1 | ... |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 2 | ... |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 3 | ... |
| ... | AVGLIFETIME | STDDEVLIFETIME | LAST_RESET |
| ... | ----- | ----- | ----- |
| ... | 691.242 | 34.322 | 2006-07-24-11.44 |
| ... | 644.740 | 22.124 | 2006-07-24-11.44 |
| ... | 612.431 | 43.347 | 2006-07-24-11.44 |
| ... | 593.451 | 28.329 | 2006-07-24-11.44 |

Example 2: The same table function can also give the highest value for average concurrency of coordinator activities running in the service class on each partition:

```

SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       CONCURRENT_ACT_TOP AS ACTTOP,
       CONCURRENT_WLO_TOP AS CONNTOP
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS_V97(CAST(NULL AS VARCHAR(128)),
        CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME, PART

```

Sample output is as follows:

| SUPERCLASS_NAME | SUBCLASS_NAME | PART | ACTTOP | CONNTOP |
|---------------------|--------------------|------|--------|---------|
| ----- | ----- | ---- | ----- | ----- |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 0 | 10 | 7 |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 1 | 0 | 0 |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 2 | 0 | 0 |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 3 | 0 | 0 |

By checking the average execution times and numbers of activities in the output of this table function, you can get a good high-level view of the load on each partition for a specific database. Any significant variations in the high-level gauges returned by this table function might indicate a change in the load on the system.

Example 3: If an activity uses thresholds with REMAP ACTIVITY TO actions, the activity might spend time in more than one service class during its lifetime. You can determine how many activities have passed through a set of service classes by looking at the ACTIVITIES_MAPPED_IN and ACTIVITIES_MAPPED_OUT columns, as shown in the following example:

```

SELECT SUBSTR(SERVICE_SUPERCLASS_NAME,1,19) AS SUPERCLASS_NAME,
       SUBSTR(SERVICE_SUBCLASS_NAME,1,18) AS SUBCLASS_NAME,
       ACTIVITIES_MAPPED_IN AS MAPPED_IN,
       ACTIVITIES_MAPPED_OUT AS MAPPED_OUT
FROM TABLE(WLM_GET_SERVICE_SUBCLASS_STATS_V97(CAST(NULL AS VARCHAR(128)),
        CAST(NULL AS VARCHAR(128)), -2)) AS SCSTATS
ORDER BY SUPERCLASS_NAME, SUBCLASS_NAME

```

Sample output is as follows:

| SUPERCLASS_NAME | SUBCLASS_NAME | MAPPED_IN | MAPPED_OUT |
|---------------------|--------------------|-----------|------------|
| ----- | ----- | ----- | ----- |
| SYSDEFAULTUSERCLASS | SYSDEFAULTSUBCLASS | 0 | 0 |
| SUPERCLASS1 | SYSDEFAULTSUBCLASS | 0 | 0 |
| SUPERCLASS1 | SUBCLASS1 | 0 | 7 |
| SUPERCLASS1 | SUBCLASS2 | 7 | 0 |

Usage notes

Some statistics are returned only if you set the COLLECT AGGREGATE ACTIVITY DATA and COLLECT AGGREGATE REQUEST DATA parameters for the corresponding service subclass to a value other than NONE.

The WLM_GET_SERVICE_SUBCLASS_STATS_V97 table function returns one row of data per service subclass and per partition. The function does not aggregate data across service classes (on a partition) or across partitions (for one or more service classes). However, you can use SQL queries to aggregate data.

The parameters are, in effect, ANDed together. That is, if you specify conflicting input parameters, such as a superclass named SUPA and a subclass named SUBB such that SUBB is not a subclass of SUPA, no rows are returned.

Information returned

Table 345. Information returned for WLM_GET_SERVICE_SUBCLASS_STATS_V97

| Column name | Data type | Description |
|---------------------------|--------------|--|
| SERVICE_SUPERCLASS_NAME | VARCHAR(128) | service_superclass_name - Service superclass name monitor element |
| SERVICE_SUBCLASS_NAME | VARCHAR(128) | service_subclass_name - Service subclass name monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |
| COORD_ACT_COMPLETED_TOTAL | BIGINT | coord_act_completed_total - Coordinator activities completed total monitor element |
| COORD_ACT_ABORTED_TOTAL | BIGINT | coord_act_aborted_total - Coordinator activities aborted total monitor element |
| COORD_ACT_REJECTED_TOTAL | BIGINT | coord_act_rejected_total - Coordinator activities rejected total monitor element |
| CONCURRENT_ACT_TOP | INTEGER | concurrent_act_top - Concurrent activity top monitor element |
| COORD_ACT_LIFETIME_TOP | BIGINT | coord_act_lifetime_top - Coordinator activity lifetime top monitor element |
| COORD_ACT_LIFETIME_AVG | DOUBLE | coord_act_lifetime_avg - Coordinator activity lifetime average monitor element |

Table 345. Information returned for WLM_GET_SERVICE_SUBCLASS_STATS_V97 (continued)

| Column name | Data type | Description |
|----------------------------|-----------|---|
| COORD_ACT_LIFETIME_STDDEV | DOUBLE | <p>Standard deviation of lifetime for coordinator activities at nesting level 0 that were associated with this service subclass since the last reset. If the COLLECT AGGREGATE ACTIVITY DATA parameter of the service class is set to NONE, the value of the column is null. Units are milliseconds.</p> <p>This standard deviation is computed from the coordinator activity lifetime histogram and may be inaccurate if the histogram is not correctly sized to fit the data. The value of -1 is returned if any values fall into the last histogram bin.</p> <p>The COORD_ACT_LIFETIME_STDDEV value of a service subclass is unaffected by activities that pass through the service subclass but are remapped to a different subclass before they are completed.</p> |
| COORD_ACT_EXEC_TIME_AVG | DOUBLE | <p>coord_act_exec_time_avg - Coordinator activities execution time average monitor element</p> |
| COORD_ACT_EXEC_TIME_STDDEV | DOUBLE | <p>Standard deviation of the execution times for coordinator activities at nesting level 0 that were associated with this service subclass since the last reset. Units are milliseconds.</p> <p>This standard deviation is computed from the coordinator activity executetime histogram and may be inaccurate if the histogram is not correctly sized to fit the data. The value of -1 is returned if any values fall into the last histogram bin.</p> <p>The execution time standard deviation of a service subclass is unaffected by activities that pass through the subclass but are remapped to a different subclass before they are completed.</p> |
| COORD_ACT_QUEUE_TIME_AVG | DOUBLE | <p>coord_act_queue_time_avg - Coordinator activity queue time average monitor element</p> |

Table 345. Information returned for WLM_GET_SERVICE_SUBCLASS_STATS_V97 (continued)

| Column name | Data type | Description |
|-----------------------------|-----------|--|
| COORD_ACT_QUEUE_TIME_STDDEV | DOUBLE | <p>Standard deviation of the queue time for coordinator activities at nesting level 0 that were associated with this service subclass since the last reset. If the COLLECT AGGREGATE ACTIVITY DATA parameter of the service class is set to NONE, the value of the column is null. Units are milliseconds.</p> <p>This standard deviation is computed from the coordinator activity queue time histogram and may be inaccurate if the histogram is not correctly sized to fit the data. The value of -1 is returned if any values fall into the last histogram bin.</p> <p>The queue time standard deviation is counted only toward the service subclass in which the activity was queued.</p> |
| NUM_REQUESTS_ACTIVE | BIGINT | Number of requests that are running in the service subclass at the time that this table function is running. |
| NUM_REQUESTS_TOTAL | BIGINT | <p>Number of requests that finished running in this service subclass since the last reset. This finished state applies to any request regardless of its membership in an activity. If the COLLECT AGGREGATE ACTIVITY DATA parameter of the service class is set to NONE, the value of the column is null.</p> <p>The NUM_REQUESTS_TOTAL value of a service subclass is unaffected by requests that pass through the service subclass but are not completed in it.</p> |
| REQUEST_EXEC_TIME_AVG | DOUBLE | request_exec_time_avg - Request execution time average monitor element |
| REQUEST_EXEC_TIME_STDDEV | DOUBLE | <p>Standard deviation of the execution times for requests that were associated with this service subclass since the last reset. Units are milliseconds. If the COLLECT AGGREGATE REQUEST DATA parameter of the service class is set to NONE, the value of this column is NULL.</p> <p>This standard deviation is computed from the request executetime histogram and may be inaccurate if the histogram is not correctly sized to fit the data. The value of -1 is returned if any values fall into the last histogram bin.</p> <p>The execution time standard deviation of a service subclass is unaffected by requests that pass through the subclass but were not completed in it.</p> |

Table 345. Information returned for WLM_GET_SERVICE_SUBCLASS_STATS_V97 (continued)

| Column name | Data type | Description |
|-------------------------|-----------|---|
| REQUEST_EXEC_TIME_TOTAL | BIGINT | Sum of the execution times for requests that were associated with this service subclass since the last reset. Units are milliseconds. If the COLLECT AGGREGATE REQUEST DATA parameter of the service class is set to NONE, the value of this column is NULL.

This total is computed from the request execution time histogram and may be inaccurate if the histogram is not correctly sized to fit the data. The value of -1 is returned if any values fall into the last histogram bin.

The execution time total of a service subclass is unaffected by requests that pass through the subclass but are not completed in it. |
| ACT_REMAPPED_IN | BIGINT | Number of activities remapped into this service subclass by a threshold REMAP ACTIVITY action since the last reset. |
| ACT_REMAPPED_OUT | BIGINT | Number of activities remapped out of this service subclass by a threshold REMAP ACTIVITY action since the last reset. |
| CONCURRENT_WLO_TOP | INTEGER | concurrent_wlo_top - Concurrent workload occurrences top monitor element |
| UOW_TOTAL_TIME_TOP | BIGINT | uow_total_time_top - UOW total time top monitor element |

WLM_GET_WORKLOAD _OCCURRENCE_ACTIVITIES _V97 - Return a list of activities

The WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97 table function returns the list of all activities that were submitted by a specified application on a specified member and have not yet been completed.

Note: This table function has been deprecated and replaced by the WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES table function.

Refer to Table 346 on page 1599 for a complete list of information that can be returned.

Syntax

```

►►WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97(—application_handle—,—————►
►—member—)—————►◄

```

The schema is SYSPROC.

Table function parameters

application_handle

An input argument of type BIGINT that specifies an application handle for which a list of activities is to be returned. If the argument is null, the data is retrieved for all the applications in the database.

member

An input argument of type INTEGER that specifies a valid member number in the same instance as the currently connected database. Specify -1 for the current member, or -2 for all members. If the null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Example 1: Activities currently running with a known application handle

After you identify the application handle, you can look up all the activities currently running in this application. For example, suppose that an administrator wants to list the activities of an application whose application handle, determined by using the **LIST APPLICATIONS** command, is 1. The administrator runs the following query:

```
SELECT SUBSTR(CHAR(COORD_PARTITION_NUM),1,5) AS COORD,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       SUBSTR(CHAR(UOW_ID),1,5) AS UOWID,
       SUBSTR(CHAR(ACTIVITY_ID),1,5) AS ACTID,
       SUBSTR(CHAR(PARENT_UOW_ID),1,8) AS PARUOWID,
       SUBSTR(CHAR(PARENT_ACTIVITY_ID),1,8) AS PARACTID,
       ACTIVITY TYPE AS ACTTYPE,
       SUBSTR(CHAR(NESTING_LEVEL),1,7) AS NESTING
FROM TABLE(WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97(1, -2)) AS WLOACTS
ORDER BY PART, UOWID, ACTID
```

Sample output from the query is as follows:

| COORD | PART | UOWID | ACTID | PARUOWID | PARACTID | ACTTYPE | NESTING |
|-------|------|-------|-------|----------|----------|----------|---------|
| 0 | 0 | 2 | 3 | - | - | CALL | 0 |
| 0 | 0 | 2 | 5 | 2 | 3 | READ_DML | 1 |

Example 2: Activities currently running on the system

The following query joins the WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97 output with the MON_GET_PKG_CACHE_STMT output on EXECUTABLE_ID to provide statement text for all the SQL activities currently running on the system:

```

SELECT t.application_handle,
       t.uow_id,
       t.activity_id,
       varchar(p.stmt_text, 256) as stmt_text
FROM table(wlm_get_workload_occurrence_activities_v97(NULL, -1)) as t,
     table(mon_get_pkg_cache_stmt(NULL, NULL, NULL, -1)) as p
WHERE t.executable_id = p.executable_id

```

Sample output is as follows:

| APPLICATION_HANDLE | UOW_ID | ACTIVITY_ID | STMT_TEXT |
|--------------------|--------|-------------|-----------------------------|
| 1 | 1 | 1 | SELECT * FROM SYSCAT.TABLES |
| 47 | 1 | 36 | INSERT INTO T1 VALUES(123) |

Information returned

Table 346. Information returned by WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97

| Column name | Data type | Description |
|---------------------|-------------|--|
| APPLICATION_HANDLE | BIGINT | application_handle - Application handle monitor element |
| MEMBER | SMALLINT | member - Database member monitor element |
| COORD_PARTITION_NUM | SMALLINT | coord_partition_num - Coordinator partition number monitor element |
| LOCAL_START_TIME | TIMESTAMP | local_start_time - Local start time monitor element |
| UOW_ID | INTEGER | uow_id - Unit of work ID monitor element |
| ACTIVITY_ID | INTEGER | activity_id - Activity ID monitor element |
| PARENT_UOW_ID | INTEGER | parent_uow_id - Parent unit of work ID monitor element |
| PARENT_ACTIVITY_ID | INTEGER | parent_activity_id - Parent activity ID monitor element |
| ACTIVITY_STATE | VARCHAR(32) | activity_state - Activity state monitor element |

Table 346. Information returned by WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97 (continued)

| Column name | Data type | Description |
|-----------------------------|-------------|---|
| ACTIVITY_STATE (continued) | VARCHAR(32) | <p>Activity state. Possible values are as follows:</p> <p>QUEUED
 The activity is queued by a workload management queuing threshold. In a partitioned database environment, this state might mean that the coordinator agent has made an RPC to the catalog partition to obtain threshold tickets and has not yet received a response. This state might indicate that the activity has been queued by a workload management queuing threshold or, if not much time has elapsed, can indicate that the activity is in the process of obtaining its tickets. To obtain a more accurate picture of whether the activity is being queued, determine what agent is working on the activity, and find out whether the EVENT_OBJECT value of the object at the catalog partition has a value of WLM_QUEUE.</p> <p>TERMINATING
 The activity has finished running and is being removed from the system.</p> |
| ACTIVITY_TYPE | VARCHAR(32) | <p>Activity type. Possible values are as follows:</p> <ul style="list-style-type: none"> • CALL • DDL • LOAD • OTHER • READ_DML • WRITE_DML |
| NESTING_LEVEL | INTEGER | nesting_level - Nesting level monitor element |
| INVOCATION_ID | INTEGER | invocation_id - Invocation ID monitor element |
| ROUTINE_ID | INTEGER | routine_id - Routine ID monitor element |
| UTILITY_ID | INTEGER | utility_id - Utility ID monitor element |
| SERVICE_CLASS_ID | INTEGER | service_class_id - Service class ID monitor element |
| DATABASE_WORK_ACTION_SET_ID | INTEGER | <p>One of the following values:</p> <ul style="list-style-type: none"> • If this activity has been categorized into a work class of database scope, the value is the ID of the work class set of which this work class is a member. • If this activity has not been categorized into a work class of database scope, the value is null. |

Table 346. Information returned by WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97 (continued)

| Column name | Data type | Description |
|----------------------------------|--------------------------|---|
| DATABASE_WORK_CLASS_ID | INTEGER | One of the following values: <ul style="list-style-type: none"> If this activity has been categorized into a work class of database scope, the value is the ID of the work class. If this activity has not been categorized into a work class of database scope, the value is null. |
| SERVICE_CLASS_WORK_ACTION_SET_ID | INTEGER | One of the following values: <ul style="list-style-type: none"> If this activity has been categorized into a work class of service class scope, the value is the ID of the work action set associated with the work class set to which the work class belongs. If this activity has not been categorized into a work class of service class scope, the value is null. |
| SERVICE_CLASS_WORK_CLASS_ID | INTEGER | One of the following values: <ul style="list-style-type: none"> If this activity has been categorized into a work class of service class scope, the value is the ID of the work class assigned to this activity. If this activity has not been categorized into a work class of service class scope, the value is null. |
| EXECUTABLE_ID | VARCHAR(32) FOR BIT DATA | executable_id - Executable ID monitor element |
| TOTAL_CPU_TIME | BIGINT | total_cpu_time - Total CPU time |
| ROWS_READ | BIGINT | rows_read - Rows read |
| ROWS_RETURNED | BIGINT | rows_returned - Rows returned |
| QUERY_COST_ESTIMATE | BIGINT | query_cost_estimate - Query cost estimate |
| DIRECT_READS | BIGINT | direct_reads - Direct reads from database |
| DIRECT_WRITES | BIGINT | direct_writes - Direct writes to database |

WLM_GET_WORKLOAD_STATS_V97 - return workload statistics

The WLM_GET_WORKLOAD_STATS_V97 table function returns workload statistics for every combination of workload name and database member number.

Note: This table function has been deprecated and replaced by the WLM_GET_WORKLOAD_STATS table function.

Refer to Table 347 on page 1603 for a complete list of information that can be returned.

Syntax

►►—WLM_GET_WORKLOAD_STATS_V97—(—workload_name—,—member—)—►►

The schema is SYSPROC.

Table function parameters

- workload_name*
An input argument of type VARCHAR(128) that specifies a workload for which the statistics are to be returned. If the argument is NULL or an empty string, statistics are returned for all workloads.
- member*
An input argument of type INTEGER that specifies the number of a member in the same instance as the currently connected database. Specify -1 for the current member, or -2 for all members. If a null value is specified, -1 is set implicitly.

Authorization

One of the following authorities is required to execute the routine:

- EXECUTE privilege on the routine
- DATAACCESS authority
- DBADM authority
- SQLADM authority
- WLMADM authority

Default PUBLIC privilege

None

Example

The following query displays statistics for workloads:

```
SELECT SUBSTR(WORKLOAD_NAME,1,18) AS WL_DEF_NAME,
       SUBSTR(CHAR(DBPARTITIONNUM),1,4) AS PART,
       COORD_ACT_LIFETIME_TOP,
       COORD_ACT_LIFETIME_AVG,
       COORD_ACT_LIFETIME_STDDEV
FROM TABLE(WLM_GET_WORKLOAD_STATS_V97(CAST(NULL AS VARCHAR(128)), -2)) AS WLSTATS
ORDER BY WL_DEF_NAME, PART
```

Sample output from the query is as follows:

| WL_DEF_NAME | PART | COORD_ACT_LIFETIME_TOP | ... |
|--------------------|------------------------|---------------------------|-----|
| ----- | | | |
| SYSDEFAULTADWORKL | 0 | -1 | ... |
| SYSDEFAULTUSERWORK | 0 | -1 | ... |
| WL1 | 0 | 2 | ... |
| ... | COORD_ACT_LIFETIME_AVG | COORD_ACT_LIFETIME_STDDEV | |
| ----- | | | |
| ... | -1.00000000000000E+000 | -1.00000000000000E+000 | |
| ... | -1.00000000000000E+000 | -1.00000000000000E+000 | |
| ... | +2.56000000000000E+000 | +6.00000000000001E-002 | |

Usage note

The function does not aggregate data across workloads, members, or service classes. However, you can use SQL queries to aggregate data.

Information returned

Table 347. Information returned by WLM_GET_WORKLOAD_STATS_V97

| Column name | Data type | Description |
|----------------------------|--------------|---|
| WORKLOAD_NAME | VARCHAR(128) | workload_name - Workload name monitor element |
| DBPARTITIONNUM | SMALLINT | dbpartitionnum - Database partition number monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |
| CONCURRENT_WLO_TOP | INTEGER | concurrent_wlo_top - Concurrent workload occurrences top monitor element |
| CONCURRENT_WLO_ACT_TOP | INTEGER | concurrent_wlo_act_top - Concurrent WLO activity top monitor element |
| COORD_ACT_COMPLETED_TOTAL | BIGINT | coord_act_completed_total - Coordinator activities completed total monitor element |
| COORD_ACT_ABORTED_TOTAL | BIGINT | coord_act_aborted_total - Coordinator activities aborted total monitor element |
| COORD_ACT_REJECTED_TOTAL | BIGINT | coord_act_rejected_total - Coordinator activities rejected total monitor element |
| WLO_COMPLETED_TOTAL | BIGINT | wlo_completed_total - Workload occurrences completed total monitor element |
| COORD_ACT_LIFETIME_TOP | BIGINT | coord_act_lifetime_top - Coordinator activity lifetime top monitor element |
| COORD_ACT_LIFETIME_AVG | DOUBLE | coord_act_lifetime_avg - Coordinator activity lifetime average monitor element |
| COORD_ACT_LIFETIME_STDDEV | DOUBLE | Standard deviation of lifetime for completed or aborted coordinator activities at nesting level 0 that are associated with this workload. Units are milliseconds. If the COLLECT AGGREGATE ACTIVITY DATA parameter of the workload is set to NONE, the value of the column is null. This standard deviation is computed from the coordinator activity lifetime histogram and may be inaccurate if the histogram is not correctly sized to fit the data. If any values fall into the last histogram bin, the value -1 is returned. |
| COORD_ACT_EXEC_TIME_AVG | DOUBLE | coord_act_exec_time_avg - Coordinator activities execution time average monitor element |
| COORD_ACT_EXEC_TIME_STDDEV | DOUBLE | Standard deviation of the execution times for completed or aborted coordinator activities at nesting level 0 that are associated with this workload. Units are milliseconds. This standard deviation is computed from the coordinator activity executetime histogram and may be inaccurate if the histogram is not correctly sized to fit the data. If any values fall into the last histogram bin, the value -1 is returned. If the COLLECT AGGREGATE ACTIVITY DATA parameter of the workload is set to NONE, the value of the column is null. |
| COORD_ACT_QUEUE_TIME_AVG | DOUBLE | coord_act_queue_time_avg - Coordinator activity queue time average monitor element |

Table 347. Information returned by WLM_GET_WORKLOAD_STATS_V97 (continued)

| Column name | Data type | Description |
|-----------------------------|-----------|---|
| COORD_ACT_QUEUE_TIME_STDDEV | DOUBLE | Standard deviation of the queue time for completed or aborted coordinator activities at nesting level 0 that are associated with this workload. Units are milliseconds. If the COLLECT AGGREGATE ACTIVITY DATA parameter of the workload is set to NONE, the value of the column is null. This standard deviation is computed from the coordinator activity queue time histogram and may be inaccurate if the histogram is not correctly sized to fit the data. If any values fall into the last histogram bin, the value -1 is returned. |
| UOW_TOTAL_TIME_TOP | BIGINT | uow_total_time_top - UOW total time top monitor element |

Index

A

- ADD CONTACT command
 - using ADMIN_CMD procedure 6
- ADD CONTACTGROUP command
 - using ADMIN_CMD procedure 7
- ADMIN_CMD procedure 4
 - commands
 - ADD CONTACT 6
 - ADD CONTACTGROUP 7
 - AUTOCONFIGURE 8
 - BACKUP DATABASE 12
 - DESCRIBE 22
 - DROP CONTACT 31
 - DROP CONTACTGROUP 31
 - EXPORT 32
 - FORCE APPLICATION 43
 - GET STMM TUNING 45
 - IMPORT 46
 - INITIALIZE TAPE 75
 - LOAD 76
 - PRUNE HISTORY/LOGFILE 122
 - QUIESCE DATABASE 124
 - QUIESCE TABLESPACES FOR TABLE 127
 - REDISTRIBUTE DATABASE PARTITION GROUP 130
 - REORG INDEXES/TABLE 137
 - RESET ALERT
 - CONFIGURATION 156
 - RESET DATABASE
 - CONFIGURATION 158
 - RESET DATABASE MANAGER
 - CONFIGURATION 160
 - REWIND TAPE 161
 - RUNSTATS 161
 - SET TAPE POSITION 177
 - UNQUIESCE DATABASE 178
 - UPDATE ALERT
 - CONFIGURATION 179
 - UPDATE CONTACT 183
 - UPDATE CONTACTGROUP 184
 - UPDATE DATABASE
 - CONFIGURATION 185
 - UPDATE DATABASE MANAGER
 - CONFIGURATION 189
 - UPDATE HEALTH NOTIFICATION CONTACT LIST 191
 - UPDATE HISTORY 191
 - UPDATE STMM TUNING 193
 - details 4
 - messages
 - removing 269
 - retrieving 215
 - ADMIN_COPY_SCHEMA procedure
 - details 195
 - ADMIN_DROP_SCHEMA procedure 199
 - ADMIN_EST_INLINE_LENGTH function 201
 - ADMIN_GET_DBP_MEM_USAGE table function 1125
 - ADMIN_GET_ENCRYPTION_INFO table function
 - details 203
 - ADMIN_GET_INDEX_COMPRESS_INFO table function 205
 - ADMIN_GET_INDEX_INFO table function 209
 - ADMIN_GET_INTRA_PARALLEL scalar function 213
 - ADMIN_GET_MEM_USAGE table function 214
 - ADMIN_GET_MSGS table function 215
 - ADMIN_GET_STORAGE_PATHS table function 217
 - ADMIN_GET_TAB_COMPRESS_INFO table function 219
 - ADMIN_GET_TAB_COMPRESS_INFO table function (deprecated) 1126
 - ADMIN_GET_TAB_COMPRESS_INFO_V97 table function 1132
 - ADMIN_GET_TAB_DICTIONARY_INFO table function 221
 - ADMIN_GET_TAB_INFO table function 224, 274
 - ADMIN_GET_TAB_INFO_V95 table function 1138
 - ADMIN_GET_TAB_INFO_V97 table function
 - details 1144
 - ADMIN_GET_TEMP_COLUMNS table function 233, 283
 - ADMIN_GET_TEMP_TABLES table function 237, 287
 - ADMIN_IS_INLINED function 240
 - ADMIN_LIST_DB_PATHS table function 1087
 - ADMIN_MOVE_TABLE procedure 242
 - ADMIN_MOVE_TABLE_UTIL procedure 266
 - ADMIN_REMOVE_MSGS procedure 269
 - ADMIN_REVALIDATE_DB_OBJECTS procedure 269
 - ADMIN_ROTATE_MASTER_KEY procedure
 - details 925
 - ADMIN_SET_INTRA_PARALLEL procedure 273
 - ADMIN_TASK_ADD procedure 290
 - ADMIN_TASK_LIST administrative view 296
 - ADMIN_TASK_REMOVE procedure 297
 - ADMIN_TASK_STATUS administrative view 299
 - ADMIN_TASK_UPDATE procedure 301
 - administrative routines
 - ADMIN_CMD procedure 4
 - routines and views 4
 - administrative task scheduler
 - defining task schedules 294
 - routines and views 290
 - ADMINTABCOMPRESSINFO administrative view (deprecated) 1126
 - ADMINTABINFO administrative view 224, 274
 - ADMINTEMPCOLUMNS administrative view 233, 283
 - ADMINTEMPTables administrative view 237, 287
 - ALTER_ROUTINE_PACKAGE procedure 990
 - ALTOBJ procedure 1075
 - AM_BASE_RPT_RECOMS table function 1150
 - AM_BASE_RPTS table function 1152
 - AM_DROP_TASK procedure 1153
 - AM_GET_LOCK_CHN_TB procedure 1154
 - AM_GET_LOCK_CHNS procedure 1155
 - AM_GET_LOCK_RPT procedure 1156
 - AM_GET_RPT procedure 1163
 - AM_SAVE_TASK procedure 1164
 - ANALYZE_LOG_SPACE procedure 1000
 - APPL_PERFORMANCE administrative view 1165
 - APPLICATION_ID scalar function 1171
 - APPLICATIONS administrative view 1167
 - audit
 - routines and views 303
 - AUDIT_ARCHIVE stored procedure and table function 303
 - AUDIT_DELIM_EXTRACT stored procedure 304
 - AUDIT_LIST_LOGS table function 305
 - AUTH_GET_INSTANCE_AUTHID scalar function 926
 - AUTH_LIST_AUTHORITIES_FOR_AUTHID table function 927
 - AUTH_LIST_GROUPS_FOR_AUTHID table function 931
 - AUTH_LIST_ROLES_FOR_AUTHID function 932
 - authorization IDs
 - built-in routines 2
 - built-in views 2
 - group membership list 931
 - instance owner 926
 - retrieving 935
 - AUTHORIZATIONIDS administrative view 935
 - AUTOCONFIGURE command
 - using ADMIN_CMD procedure 8
 - AUTOMAINT_GET_POLICY stored procedure 306
 - AUTOMAINT_GET_POLICYFILE stored procedure 308
 - AUTOMAINT_SET_POLICY stored procedure 309

AUTOMAINT_SET_POLICYFILE stored
procedure 311
automatic maintenance
routines and views 306

B

BACKUP DATABASE command
using ADMIN_CMD procedure 12
BP_HITRATIO administrative view 1172
BP_READ_IO administrative view 1174
BP_WRITE_IO administrative view 1176
built-in routines
authorizations 2
coding practices 1
summary 4
built-in views
ADMIN_TASK_LIST 296
ADMIN_TASK_STATUS 299
ADMINTABCOMPRESSINFO
(deprecated) 1126
ADMINTABINFO 224, 274
ADMINTEMPCOLUMNS 233, 283
ADMINTEMPTABLES 237, 287
APPL_PERFORMANCE
details 1165
APPLICATIONS 1167
AUTHORIZATIONIDS
details 935
authorizations 2
BP_HITRATIO
details 1172
BP_READ_IO
details 1174
BP_WRITE_IO
details 1176
coding practices 1
CONTACTGROUPS 1080
CONTACTS 1082
CONTAINER_UTILIZATION 1178
DB_HISTORY
details 1083
DB2_CF
details 376
DB2_CLUSTER_HOST_STATE
details 373
DB2_INSTANCE_ALERTS
details 375
DB2_MEMBER
details 376
DBCFCG 363
DBMCFG 367
DBPATHS 1087
ENV_CF_SYS_RESOURCES 381
ENV_FEATURE_INFO 383
ENV_INST_INFO 398
ENV_PROD_INFO 400
ENV_SYS_INFO 401
ENV_SYS_RESOURCES 1182
LOCKS_HELD 1236
LOCKWAIT 1239
LOG_UTILIZATION 1242
LONG_RUNNING_SQL
details 1244
MON_BP_UTILIZATION 422
MON_CONNECTION_SUMMARY 430
MON_CURRENT_SQL 435

built-in views (*continued*)
MON_CURRENT_UOW 436
MON_DB_SUMMARY 438
MON_LOCKWAITS 441
MON_PKG_CACHE_SUMMARY 443
MON_SERVICE_SUBCLASS_SUMMARY 445
MON_TBSP_UTILIZATION 449
MON_TRANSACTION_LOG_UTILIZATION 449
MON_WORKLOAD_SUMMARY 455
NOTIFICATIONLIST 1093
OBJECTOWNERS
details 936
overview 1
PDLOGMSG_LAST24HOURS 1102
PRIVILEGES
details 937
QUERY_PREP_COST
details 938
REG_VARIABLES 1247
SNAPAGENT 1250, 1430
SNAPAGENT_MEMORY_POOL 1254,
1433
SNAPAPPL 1272, 1446
SNAPAPPL_INFO 1258, 1438
SNAPBP 1287, 1455
SNAPBP_PART 1293, 1460
SNAPCONTAINER 1301, 1464
SNAPDB 1308, 1468, 1518
SNAPDB_MEMORY_POOL 1320,
1480
SNAPDBM 1334, 1484
SNAPDBM_MEMORY_POOL 1339,
1489
SNAPDETAILLOG 1345, 1493
SNAPDYN_SQL 940, 964
SNAPFCM 1355, 1496
SNAPFCM_PART 1358, 1499
SNAPHADR 1361, 1503
SNAPLOCK 1366, 1507
SNAPLOCKWAIT 1371, 1512
SNAPSTMT 1382, 1534
SNAPSTORAGE_PATHS 1390, 1541
SNAPSUBSECTION 946, 970
SNAPSWITCHES 950, 974
SNAPTAB 1393, 1544
SNAPTAB_REORG 954, 978
SNAPTbsp 1399, 1548
SNAPTbsp_PART 1405, 1553
SNAPTbsp_QUIESCER 1419, 1559
SNAPTbsp_RANGE 1377, 1563
SNAPUTIL 1424, 1530, 1568
SNAPUTIL_PROGRESS 960, 984
summary 4
table functions comparison 3
TBSP_UTILIZATION 1572
TOP_DYNAMIC_SQL
details 988

C

CANCEL_WORK stored procedure 318
CAPTURE_STORAGE_MGMT_INFO
procedure 1009
commands
ADD CONTACT 6
ADD CONTACTGROUP 7
AUTOCONFIGURE 8

commands (*continued*)
BACKUP DATABASE 12
calling from procedure 4, 1013
DESCRIBE
details 22
DROP CONTACT 31
DROP CONTACTGROUP 31
EXPORT 32
FORCE APPLICATION 43
GET STMM TUNING 45
IMPORT 46
INITIALIZE TAPE 75
LOAD 76
PRUNE HISTORY/LOGFILE 122
PRUNE LOGFILE
details 122
QUIESCE DATABASE 124
QUIESCE TABLESPACES FOR
TABLE 127
REDISTRIBUTE DATABASE
PARTITION GROUP 130
REORG INDEXES/TABLE 137
RESET ALERT
CONFIGURATION 156
RESET DATABASE
CONFIGURATION 158
RESET DATABASE MANAGER
CONFIGURATION 160
REWIND TAPE 161
RUNSTATS 161
SET TAPE POSITION 177
UNQUIESCE DATABASE 178
UPDATE ALERT
CONFIGURATION 179
UPDATE CONTACT 183
UPDATE CONTACTGROUP 184
UPDATE DATABASE
CONFIGURATION 185
UPDATE DATABASE MANAGER
CONFIGURATION 189
UPDATE HEALTH NOTIFICATION
CONTACT LIST 191
UPDATE HISTORY 191
UPDATE STMM TUNING 193
common SQL API stored procedures
complete mode 315
filtering output 316
overview 312
signature 312
stored procedures 313
XML input documents 314
XML message documents 317
XML output files 315
COMPILATION_ENV table
function 1078
complete mode 315
configuration
routines and views 363
configuration parameters
database
retrieving values 363
getting values 331
setting with SET_CONFIG
procedure 353
CONTACTGROUPS administrative
view 1080

- contacts
 - retrieving contact group lists 1080
 - retrieving contact lists 1082
- CONTACTS administrative view 1082
- CONTAINER_UTILIZATION administrative view 1178
- CREATE_STORAGE_MGMT_TABLES procedure 1011

D

- data distribution
 - routines and views 1000
- database environment
 - routines and views 381
- database manager configuration
 - parameters
 - retrieving values 367
- database paths
 - retrieving 1087
- DB_GET_CFG table function 363
- DB_HISTORY administrative view
 - details 1083
- DB_MEMBERS table function 371
- DB_PARTITIONS table function 1180
- Db2 pureScale instances
 - hosts
 - information 371
- DB2 Text Search
 - routines and views 1013
- DB2_CF administrative view
 - details 376
- DB2_CLUSTER_HOST_STATE administrative view
 - details 373
- DB2_GET_CLUSTER_HOST_STATE table function
 - details 373
- DB2_GET_INSTANCE_INFO table function
 - details 376
- DB2_INSTANCE_ALERTS administrative view
 - details 375
- DB2_MEMBER administrative view
 - details 376
- DBCFCG administrative view 363
- DBMCFG administrative view 367
- DBPATHS administrative view 1087
- deprecated functionality
 - administrative views
 - SNAPDB_MEMORY_POOL 1320, 1480
 - SNAPSTORAGE_PATHS 1390, 1541
 - SNAPTAB 1393, 1544
- procedures
 - GET_DB_CONFIG 1188
 - HEALTH_CONT_HI 1191
 - HEALTH_CONT_HI_HIS 1193
 - HEALTH_CONT_INFO 1195
 - HEALTH_DB_HI 1197
 - HEALTH_DB_HI_HIS 1200
 - HEALTH_DB_HIC 1204
 - HEALTH_DB_HIC_HIS 1206
 - HEALTH_DB_INFO 1209
 - HEALTH_DBM_HI 1211

- deprecated functionality (*continued*)
- procedures (*continued*)

- HEALTH_DBM_HI_HIS 1213
- HEALTH_DBM_INFO 1215
- HEALTH_GET_ALERT_ACTION_CFGGEN 1217
- HEALTH_GET_ALERT_CFG 1220
- HEALTH_GET_IND_DEFINITION 1223
- HEALTH_HI_REC 1225
- HEALTH_TBS_HI 1227
- HEALTH_TBS_HI_HIS 1230
- HEALTH_TBS_INFO 1234
- SYSINSTALLROUTINES 1572
- WLM_GET_WORKLOAD_OCCURRENCES_V97 1597
- WLM_GET_WORKLOAD_STATS_V97 1600
- SQL administrative routines 1119
- table functions
 - ADMIN_GET_TAB_COMPRESS_INFO 1126
 - ADMIN_GET_TAB_COMPRESS_INFO_V97 1126
 - GET_DBM_CONFIG 1190
 - SNAP_GET_APPL_V95 1280
 - SNAP_GET_BP_V95 1297
 - SNAP_GET_CONTAINER_V91 1305
 - SNAP_GET_DB_MEMORY_POOL 1320, 1480
 - SNAP_GET_DB_V97 1324
 - SNAP_GET_DBM_V95 1342
 - SNAP_GET_DETAILLOG_V91 1349
 - SNAP_GET_DYN_SQL_V95 1351
 - SNAP_GET_STO_PATHS 1388
 - SNAP_GET_STORAGE_PATHS_V97 1541
 - SNAP_GET_TAB 1393, 1544
 - SNAP_GET_TAB_V91 1397
 - SNAP_GET_TBSP_PART_V97 1411
 - SNAP_GET_TBSP_V91 1415
 - WLM_GET_SERVICE_CLASS_AGENTS_V97 1581
 - WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97 1588
 - WLM_GET_SERVICE_SUBCLASS_STATS_V97 1590
- DESCRIBE command
 - details 22
- Design Advisor
 - DESIGN_ADVISOR stored procedure 324
- DESIGN_ADVISOR procedure 324
- DROP CONTACT command
 - details
 - using ADMIN_CMD 31
- DROP CONTACTGROUP command
 - details
 - using ADMIN_CMD 31
- DROP_STORAGE_MGMT_TABLES procedure 1012

E

- ENV_CF_SYS_RESOURCES administrative view
 - details 381
- ENV_FEATURE_INFO administrative view 383
- ENV_GET_DB2_EDU_SYSTEM_RESOURCES table function 384
- ENV_GET_DB2_SYSTEM_RESOURCES 387
- ENV_GET_NETWORK_RESOURCES 388
- ENV_GET_REG_VARIABLES 389

- ENV_GET_ROLLING_UPDATE_STATUS table function 392
- ENV_GET_SYS_RESOURCES table function 1185
- ENV_GET_SYSTEM_RESOURCES table function 395
- ENV_INST_INFO administrative view 398
- ENV_PROD_INFO administrative view 400
- ENV_SYS_INFO administrative view 401
- ENV_SYS_RESOURCES administrative view 1182
- error messages
 - retrieving 1126
- ERRM scalar functions 1115
- EVMON_FORMAT_UE_TO_TABLES procedure
 - details 462
- EVMON_FORMAT_UE_TO_XML table function
 - details 471
- EVMON_UPGRADE_TABLES procedure 474
- explain
 - routines and views 402
- EXPLAIN_FORMAT_STATS scalar function 405
- EXPLAIN_FROM_ACTIVITY procedure 410
- EXPLAIN_FROM_CATALOG procedure 413
- EXPLAIN_FROM_DATA procedure 415
- EXPLAIN_FROM_SECTION procedure 418
- EXPLAIN_GET_MSGS table function 1540
- EXPORT command
 - details
 - using ADMIN_CMD 32
- extents
 - movement status 651

F

- FORCE APPLICATION command
 - using ADMIN_CMD 43
- functions
 - scalar
 - ADMIN_GET_INTRA_PARALLEL 213
 - APPLICATION_ID 1171
 - AUTH_GET_INSTANCE_AUTHID 926
 - EXPLAIN_FORMAT_STATS 405
 - GET_ROUTINE_OPTS 993
 - MON_GET_APPLICATION_HANDLE 565
 - MON_GET_APPLICATION_ID 566
 - MQPUBLISH 904
 - MQREAD 906
 - MQREADCLOB 911
 - MQRECEIVE 912
 - MQRECEIVECLOB 919
 - MQSEND 920
 - MQSUBSCRIBE 922
 - MQUNSUBSCRIBE 923

| | | |
|-----------------------------------|--------------------------------------|----------------------------------|
| functions (continued) | functions (continued) | functions (continued) |
| scalar (continued) | table (continued) | table (continued) |
| SQLERRM 1115 | HEALTH_GET_ALERT_ACTION_CFG 1217 | MQRECEIVEALLCLOB 916 |
| stored procedures | HEALTH_GET_ALERT_CFG 1220 | overview 1 |
| SYSTS_ALTER 1015 | HEALTH_GET_IND_DEFINITION 1223 | PD_GET_DIAG_HIST 1094 |
| SYSTS_CLEANUP 1021 | HEALTH_TBS_HI 1227 | PD_GET_LOG_MSGS 1102 |
| SYSTS_CLEAR_COMMANDLOCKS 1022 | HEALTH_TBS_HI_HIS 1230 | REG_LIST_VARIABLES table |
| SYSTS_CLEAR_EVENTS 1024 | HEALTH_TBS_INFO 1234 | function 1248 |
| SYSTS_DROP 1041 | MON_GET_ACTIVITY 514 | SNAP_GET_AGENT 1250, 1430 |
| SYSTS_ENABLE 1043 | MON_GET_ACTIVITY_DETAILS 533 | SNAP_GET_AGENT_MEMORY_POOL 1254, |
| SYSTS_UPDATE 1045 | MON_GET_AGENT 551 | 1433 |
| SYSTS_UPGRADE_CATALOG 1050 | MON_GET_AUTO_MAINT_QUEUE 566 | SNAP_GET_APPL 1272, 1446 |
| SYSTS_UPGRADE_INDEX 1051 | MON_GET_AUTO_RUNSTATS_QUEUE 566 | SNAP_GET_APPL_INFO 1258, |
| summary 4 | MON_GET_BUFFERPOOL 571 | 1438 |
| table | MON_GET_CF 580 | SNAP_GET_APPL_INFO_V95 1266 |
| ADMIN_GET_ENCRYPTION_INFO 203 | MON_GET_CF_CMD 582 | SNAP_GET_APPL_V95 1280 |
| ADMIN_GET_MSGS 215 | MON_GET_CF_WAIT_TIME 585 | SNAP_GET_BP 1287, 1455 |
| ADMIN_GET_STORAGE_PATHS 217 | MON_GET_CONNECTION 588 | SNAP_GET_BP_PART 1293, 1460 |
| ADMIN_GET_TAB_COMPRESS_INFO | MON_GET_CONNECTION_DETAILS 603 | SNAP_GET_BP_V95 1297 |
| (deprecated) 1126 | MON_GET_CONTAINER 616 | SNAP_GET_CONTAINER 1301, |
| ADMIN_GET_TAB_COMPRESS_INFO_V97 | MON_GET_DATABASE 619 | 1464 |
| ADMIN_GET_TAB_INFO 224, | MON_GET_DATABASE_DETAILS 636 | SNAP_GET_CONTAINER_V91 |
| 274 | MON_GET_EXTENDED_LATCH_WAIT 650 | (deprecated) 1305 |
| ADMIN_GET_TAB_INFO_V95 1138 | MON_GET_EXTENT_MOVEMENT_STATUS 651 | SNAP_GET_DB_MEMORY_POOL 1320, |
| ADMIN_GET_TAB_INFO_V97 1144 | MON_GET_FCM 652 | 1480 |
| ADMIN_GET_TEMP_COLUMNS 233, | MON_GET_FCM_CONNECTION_LIST 654 | SNAP_GET_DB_V97 1324 |
| 283 | MON_GET_GROUP_BUFFERPOOL 655 | SNAP_GET_DBM 1334, 1484 |
| ADMIN_GET_TEMP_TABLES 237, | MON_GET_HADR 656 | SNAP_GET_DBM_MEMORY_POOL 1339, |
| 287 | MON_GET_INDEX 662 | 1489 |
| ADMIN_IS_INLINED 240 | MON_GET_INDEX_USAGE_LIST 665 | SNAP_GET_DBM_V95 1342 |
| ADMIN_IS_INLINED_LENGTH 201 | MON_GET_INSTANCE 668 | SNAP_GET_DETAIL_LOG 1345, |
| ADMIN_LIST_DB_PATHS 1087 | MON_GET_LATCH table | 1493 |
| AM_BASE_RPT_RECOMS 1150 | function 670 | SNAP_GET_DETAIL_LOG_V91 1349 |
| AM_BASE_RPTS 1152 | MON_GET_PAGE_ACCESS 684 | SNAP_GET_DYN_SQL 940, 964 |
| AUDIT_ARCHIVE 303 | MON_GET_PKG_CACHE_STMT 686 | SNAP_GET_DYN_SQL_V95 1351 |
| AUDIT_LIST_LOGS 305 | MON_GET_QUEUE_STATS 712 | SNAP_GET_FCM 1355, 1496 |
| AUTH_LIST_AUTHORITIES_FOR_AUTHID | MON_GET_ROUTINE 717 | SNAP_GET_FCM_PART 1358, |
| AUTH_LIST_GROUPS_FOR_AUTHID 931 | MON_GET_ROUTINE_DETAILS 733 | 1499 |
| AUTH_LIST_ROLES_FOR_AUTHID 932 | MON_GET_ROUTINE_EXEC_LIST 748 | SNAP_GET_HADR 1361, 1503 |
| built-in views comparison 3 | MON_GET_RTS_RQST 752 | SNAP_GET_LOCK 1366, 1507 |
| COMPILATION_ENV 1078 | MON_GET_SECTION 754 | SNAP_GET_LOCKWAIT 1371, |
| DB_GET_CFG 363 | MON_GET_SECTION_OBJECT 755 | 1512 |
| DB_MEMBERS 371 | MON_GET_SECTION_ROUTINE 758 | SNAP_GET_STMT 1382, 1534 |
| DB_PARTITIONS 1180 | MON_GET_SERVERLIST 760 | SNAP_GET_STO_PATHS |
| deprecated 1119 | MON_GET_SERVICE_SUBCLASS 762 | (deprecated) 1388 |
| ENV_GET_DB2_EDU_SYSTEM_RESOURCES | MON_GET_SERVICE_SUBCLASS_DETAILS 779 | SNAP_GET_STORAGE_PATHS_V97 1390, |
| ENV_GET_DB2_SYSTEM_RESOURCES 387 | MON_GET_SERVICE_SUBCLASS_STATS 792 | 1541 |
| ENV_GET_NETWORK_RESOURCES 388 | MON_GET_SERVICE_SUPERCLASS_STATS 798 | SNAP_GET_SUBSECTION 946, |
| ENV_GET_REG_VARIABLES 389 | MON_GET_TABLE 800 | 970 |
| ENV_GET_ROLLING_UPDATE_STATUS 392 | MON_GET_TABLE_USAGE_LIST 823 | SNAP_GET_SWITCHES 950, 974 |
| ENV_GET_SYSTEM_RESOURCES 395 | MON_GET_TABLESPACE 805 | SNAP_GET_TAB 1393, 1544 |
| EVMON_FORMAT_UE_TO_XML 471 | MON_GET_TABLESPACE QUIESCER 817 | SNAP_GET_TAB_REORG 954, |
| EXPLAIN_GET_MSGS 402 | MON_GET_TABLESPACE_RANGE 820 | 978 |
| GET_DB_CONFIG 1188 | MON_GET_TRANSACTION_LOG 827 | SNAP_GET_TAB_V91 |
| GET_DBM_CONFIG 1190 | MON_GET_UNIT_OF_WORK 829 | (deprecated) 1397 |
| HEALTH_CONT_HI 1191 | MON_GET_UNIT_OF_WORK_DETAILS 845 | SNAP_GET_TBSP 1399, 1548 |
| HEALTH_CONT_HI_HIS 1193 | MON_GET_USAGE_LIST_STATUS 862 | SNAP_GET_TBSP_PART |
| HEALTH_CONT_INFO 1195 | MON_GET_UTILITY 863 | (deprecated) 1405, 1553 |
| HEALTH_DB_HI 1197 | MON_GET_WORK_ACTION_SET_STATS 895 | SNAP_GET_TBSP_PART_V97 1411 |
| HEALTH_DB_HI_HIS 1200 | MON_GET_WORKLOAD 866 | SNAP_GET_TBSP QUIESCER 1419, |
| HEALTH_DB_HIC 1204 | MON_GET_WORKLOAD_DETAILS 880 | 1559 |
| HEALTH_DB_HIC_HIS 1206 | MON_GET_WORKLOAD_STATS 892 | SNAP_GET_TBSP_RANGE 1377, |
| HEALTH_DB_INFO 1209 | MON_SAMPLE_SERVICE_CLASS_METRICS 893 | 893 |
| HEALTH_DBM_HI 1211 | MON_SAMPLE_WORKLOAD_METRICS 908 | SNAP_GET_TBSP_V91 |
| HEALTH_DBM_HI_HIS 1213 | MQREADALL 907 | (deprecated) 1415 |
| HEALTH_DBM_INFO 1215 | MQREADALLCLOB 909 | SNAP_GET_UTIL 1424, 1530, |
| | MQRECEIVEALL 914 | 1568 |

functions (continued)

table (continued)

SNAP_GET_UTIL_PROGRESS 960, 984
summary 4
WLM_GET_ACTIVITY_DETAILS 1575
WLM_GET_CONN_ENV 1061
WLM_GET_QUEUE_STATS 1063
WLM_GET_SERVICE_CLASS_AGENTS 1064
WLM_GET_SERVICE_CLASS_AGENTS 1064
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES 1064
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES_V97 1588
WLM_GET_SERVICE_SUBCLASS_STATE 1067
WLM_GET_SERVICE_SUBCLASS_STATE 1067
WLM_GET_SERVICE_SUPERCLASS_STATE 1068
WLM_GET_WORK_ACTION_SET_STATE 1068
WLM_GET_WORKLOAD_OCCURRENCES_ACTIVITIES 1069
WLM_GET_WORKLOAD_OCCURRENCES_ACTIVITIES_V97 1597
WLM_GET_WORKLOAD_STATS 1069
WLM_GET_WORKLOAD_STATS_V97 1600
table functions
ENV_GET_SYS_RESOURCES table function 1185
MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW 486
MON_FORMAT_XML_METRICS_BY_ROW 490
MON_FORMAT_XML_TIMES_BY_ROW 504
MON_FORMAT_XML_WAIT_TIMES_BY_ROW 510
MON_GET_PKG_CACHE_STMT_DETAILS 700
MON_GET_REBALANCE_STATUS 715
SNAP_GET_DB 1308, 1468, 1518

G

GENERATE_DISTFILE procedure 1002
GET STMM TUNING command 45
GET_CONFIG stored procedure 331
GET_DB_CONFIG table function 1188
GET_DBM_CONFIG table function 1190
GET_DBSIZE_INFO procedure 1091
GET_MESSAGE stored procedure 339
GET_ROUTINE_NAME procedure stored procedure 991
GET_ROUTINE_OPTS scalar function 993
GET_ROUTINE_SAR procedure 994
GET_SWRD_SETTINGS procedure 1003
GET_SYSTEM_INFO stored procedure 346
groups
retrieving group membership 931

H

health alerts
alert action configuration 1217
alert configuration 1220
health indicators
retrieving definitions 1223
HEALTH_CONT_HI table function 1191
HEALTH_CONT_HI_HIS table function 1193

HEALTH_CONT_INFO table function 1195
HEALTH_DB_HI table function 1197
HEALTH_DB_HI_HIS table function 1200
HEALTH_DB_HIC table function 1204
HEALTH_DB_HIC_HIS table function 1206
HEALTH_DB_INFO table function 1209
HEALTH_DBM_HI table function 1211
HEALTH_DBM_HI_HIS table function 1213
HEALTH_DBM_INFO table function 1215
HEALTH_GET_ALERT_ACTION_CFG 1217
HEALTH_GET_ALERT_ACTION_CFG 1217
HEALTH_GET_ALERT_CFG table 1220
HEALTH_GET_IND_DEFINITION table function 1223
HEALTH_HI_REC procedure 1225
HEALTH_TBS_HI table function 1227
HEALTH_TBS_HI_HIS table function 1230
HEALTH_TBS_INFO table function 1234
history file
retrieving information 1083
INITIALIZE TAPE command
using ADMIN_CMD procedure 75
installation
license information 383, 1182
retrieving database product information 400
instance owner authorization ID obtaining 926
instances
current
retrieving information 398

L

latch waits
MON_GET_EXTENDED_LATCH_WAIT table function 650
LOAD command
details
using ADMIN_CMD procedure 76
LOCKS_HELD administrative view 1236
LOCKWAIT administrative view 1239
LOG_UTILIZATION administrative view 1242
LONG_RUNNING_SQL administrative view 1244

M

miscellaneous routines and views 1075
MON_BP_UTILIZATION administrative view 422
MON_CAPTURE_ACTIVITY_IN_PROGRESS stored procedure 479
MON_COLLECT_STATS stored procedure 481
MON_CONNECTION_SUMMARY administrative view 430
MON_CURRENT_SQL administrative view 435
MON_CURRENT_UOW administrative view 436
MON_DB_SUMMARY administrative view 438
MON_FORMAT_LOCK_NAME table function 483
MON_FORMAT_XML_COMPONENT_TIMES_BY_ROW table function details 486
MON_FORMAT_XML_METRICS_BY_ROW table function details 490
MON_FORMAT_XML_TIMES_BY_ROW table function details 504
MON_FORMAT_XML_WAIT_TIMES_BY_ROW table function details 510
MON_GET_ACTIVITY table function 514
MON_GET_ACTIVITY_DETAILS table function 533
MON_GET_AGENT table function 551
MON_GET_APPL_LOCKWAIT table function 561
MON_GET_APPLICATION_HANDLE scalar function 565
MON_GET_APPLICATION_ID scalar function 566
MON_GET_AUTO_MAINT_QUEUE table function 566
MON_GET_AUTO_RUNSTATS_QUEUE table function 569
MON_GET_BUFFERPOOL table function 571
MON_GET_CF table function details 580
MON_GET_CF_CMD table function 582
MON_GET_CF_WAIT_TIME table function 585
MON_GET_CONNECTION table function 588
MON_GET_CONNECTION_DETAILS table function 603
MON_GET_CONTAINER table function 616
MON_GET_DATABASE table function 619
MON_GET_DATABASE_DETAILS table function 636
MON_GET_EXTENDED_LATCH_WAIT table function 650
MON_GET_EXTENT_MOVEMENT_STATUS table function 651
MON_GET_FCM table function 652

MON_GET_FCM_CONNECTION_LIST
 table function 654
 MON_GET_GROUP_BUFFERPOOL table
 function
 details 655
 MON_GET_HADR table function
 details 656
 MON_GET_INDEX table function 662
 MON_GET_INDEX_USAGE_LIST table
 function 665
 MON_GET_INSTANCE table
 function 668
 MON_GET_LATCH table function 670
 MON_GET_LOCKS table function 674
 MON_GET_MEMORY_POOL table
 function 679
 MON_GET_MEMORY_SET table
 function 682
 MON_GET_PAGE_ACCESS
 details 684
 MON_GET_PKG_CACHE_STMT table
 function
 details 686
 MON_GET_PKG_CACHE_STMT_DETAILS
 table function
 details 700
 MON_GET_QUEUE_STATS table
 function 712
 MON_GET_REBALANCE_STATUS table
 function
 details 715
 MON_GET_ROUTINE table
 function 717
 MON_GET_ROUTINE_DETAILS table
 function 733
 MON_GET_ROUTINE_EXEC_LIST table
 function 748
 MON_GET_RTS_RQST table
 function 752
 MON_GET_SECTION table function 754
 MON_GET_SECTION_OBJECT table
 function 755
 MON_GET_SECTION_ROUTINE table
 function 758
 MON_GET_SERVERLIST table
 function 760
 MON_GET_SERVICE_SUBCLASS table
 function 762
 MON_GET_SERVICE_SUBCLASS_DETAILS
 table function 779
 MON_GET_SERVICE_SUBCLASS_STATS
 table function 792
 MON_GET_SERVICE_SUPERCLASS_STATS
 table function 798
 MON_GET_TABLE table function 800
 MON_GET_TABLE_USAGE_LIST table
 function 823
 MON_GET_TABLESPACE table
 function 805
 MON_GET_TABLESPACE QUIESCE
 table function 817
 MON_GET_TABLESPACE_RANGE table
 function 820
 MON_GET_TRANSACTION_LOG table
 function 827
 MON_GET_UNIT_OF_WORK table
 function 829

MON_GET_UNIT_OF_WORK_DETAILS
 table function 845
 MON_GET_USAGE_LIST_STATUS table
 function 862
 MON_GET_UTILITY table function 863
 MON_GET_WORK_ACTION_SET_STATS
 table function 895
 MON_GET_WORKLOAD table
 function 866
 MON_GET_WORKLOAD_DETAILS table
 function 880
 MON_GET_WORKLOAD_STATS table
 function 892
 MON_INCREMENT_INTERVAL_ID
 stored procedure 897
 MON_LOCKWAITS administrative
 view 441
 MON_PKG_CACHE_SUMMARY
 administrative view 443
 MON_SAMPLE_SERVICE_CLASS_METRICS
 table function 897
 MON_SAMPLE_WORKLOAD_METRICS
 table function 901
 MON_SERVICE_SUBCLASS_SUMMARY
 administrative view 445
 MON_TBSP_UTILIZATION
 administrative view 449
 MON_TRANSACTION_LOG_UTILIZATION
 administrative view 454
 MON_WORKLOAD_SUMMARY
 administrative view 455
 monitoring
 routines 459
 views 421
 MQPUBLISH scalar function 904
 MQREAD scalar function 906
 MQREADALL table function 907
 MQREADALLCLOB table function 909
 MQREADCLOB scalar function 911
 MQRECEIVE scalar function 912
 MQRECEIVEALL table function 914
 MQRECEIVEALLCLOB table
 function 916
 MQRECEIVECLOB scalar function 919
 MQSEND scalar function 920
 MQSUBSCRIBE scalar function 922
 MQUNSUBSCRIBE scalar function 923

N

notification lists
 retrieving contact list 1093
 notification log messages
 retrieving 1102
 NOTIFICATIONLIST administrative
 view 1093

O

OBJECTOWNERS administrative
 view 936
 objects
 retrieving ownership 936
 online table moves
 ADMIN_MOVE_TABLE
 procedure 242

online table moves (*continued*)
 ADMIN_MOVE_TABLE_UTIL
 procedure 266

P

packages
 rebinding
 REBIND_ROUTINE_PACKAGE
 procedure 996
 routines and views 990
 PD_GET_DIAG_HIST table
 function 1094
 PD_GET_LOG_MSGS table
 function 1102
 PDLOGMSG_LAST24HOURS
 administrative view 1102
 privileges
 information about granted
 PRIVILEGES administrative
 view 937
 PRIVILEGES administrative view 937
 problem determination
 notification log messages 1102
 procedures
 ADMIN_CMD
 details 4
 ADMIN_COPY_SCHEMA 195
 ADMIN_DROP_SCHEMA 199
 ADMIN_MOVE_TABLE_UTIL 266
 ADMIN_REMOVE_MSGS 269
 ADMIN_REVALIDATE_DB_OBJECTS 269
 ADMIN_ROTATE_MASTER_KEY 925
 ADMIN_SET_INTRA_PARALLEL 273
 ADMIN_TASK_ADD 290
 ADMIN_TASK_REMOVE 297
 ADMIN_TASK_UPDATE 301
 ALTER_ROUTINE_PACKAGE 990
 ALTOBJ 1075
 AM_DROP_TASK 1153
 AM_GET_LOCK_CHN_TB 1154
 AM_GET_LOCK_CHNS 1155
 AM_GET_LOCK_RPT 1156
 AM_GET_RPT 1163
 AM_SAVE_TASK 1164
 ANALYZE_LOG_SPACE 1000
 AUDIT_ARCHIVE 303
 AUDIT_DELIM_EXTRACT 304
 AUTOMAINT_GET_POLICY 306
 AUTOMAINT_GET_POLICYFILE 308
 AUTOMAINT_SET_POLICY 309
 AUTOMAINT_SET_POLICYFILE 311
 CANCEL_WORK 318
 CAPTURE_STORAGEMGMT_INFO 1009
 common SQL API
 overview 312
 XPath expressions for filtering
 output 316
 CREATE_STORAGEMGMT_TABLES 1011
 deprecated 1119
 DESIGN_ADVISOR 324
 DROP_STORAGEMGMT_TABLES 1012
 EVMON_FORMAT_UE_TO_TABLES 462
 EVMON_UPGRADE_TABLES 474
 EXPLAIN_FROM_ACTIVITY 410
 EXPLAIN_FROM_CATALOG 413
 EXPLAIN_FROM_DATA 415

procedures (continued)

EXPLAIN_FROM_SECTION 418
 GENERATE_DISTFILE 1002
 GET_CONFIG 331
 GET_DBSIZE_INFO 1091
 GET_MESSAGE 339
 GET_ROUTINE_NAME
 procedure 991
 GET_ROUTINE_SAR 994
 GET_SWRD_SETTINGS 1003
 GET_SYSTEM_INFO 346
 HEALTH_HI_REC 1225
 MON_CAPTURE_ACTIVITY_IN_PROGRESS 1054
 MON_COLLECT_STATS 481
 MON_INCREMENT_INTERVAL_ID 897
 PUT_ROUTINE_SAR 995
 REBIND_ROUTINE_PACKAGE 996
 REORGCHK_IX_STATS 1110
 REORGCHK_TB_STATS 1112
 SET_CONFIG 353
 SET_MAINT_MODE_RECORD_NO_TEMPORALHISTORY 1054
 SET_ROUTINE_OPTS 999
 SET_SWRD_SETTINGS 1005
 SNAP_WRITE_FILE 1428
 SQL
 summary 4
 STEPWISE_REDISTRIBUTE_DBPG 1008
 SYSINSTALLOBJECTS 1118
 SYSINSTALLROUTINES 1572
 SYSTS_ADMIN_CMD 1013
 SYSTS_CONFIGURE 1026
 SYSTS_CREATE 1029
 WLM_ALTER_MEMBER_SUBSET 1055
 WLM_CANCEL_ACTIVITY 1053
 WLM_CAPTURE_ACTIVITY_IN_PROGRESS 1054
 WLM_COLLECT_STATS 1055
 WLM_CREATE_MEMBER_SUBSET 1058
 WLM_DROP_MEMBER_SUBSET 1061
 WLM_SET_CLIENT_INFO 1070
 WLM_SET_CONN_ENV 1072
 procedurestabl
 ADMIN_MOVE_TABLE 242
 PRUNE HISTORY/LOGFILE command
 using ADMIN_CMD procedure 122
 PUT_ROUTINE_SAR procedure 995

Q

QUERY_PREP_COST administrative
 view 938
 QUIESCE DATABASE command 124
 QUIESCE TABLESPACES FOR TABLE
 command
 using ADMIN_CMD procedure 127

R

REBIND_ROUTINE_PACKAGE
 procedure 996
 REDISTRIBUTE DATABASE PARTITION
 GROUP command
 using ADMIN_CMD procedure 130
 redistribution of data
 procedures 1000, 1002, 1003, 1005,
 1008

REG_LIST_VARIABLES table
 function 1248
 REG_VARIABLES administrative
 view 1247
 registry variables
 retrieving settings 389, 1247
 REORG INDEXES command
 using ADMIN_CMD procedure 137
 REORG TABLE command
 using ADMIN_CMD procedure 137
 REORGCHK_IX_STATS procedure 1110
 REORGCHK_TB_STATS procedure 1112
 RESET ALERT CONFIGURATION
 command
 using ADMIN_CMD procedure 156
 RESET DATABASE CONFIGURATION
 command
 using ADMIN_CMD procedure 158
 RESET DATABASE MANAGER
 CONFIGURATION command
 using ADMIN_CMD procedure 160
 revalidation
 procedures 269
 REWIND TAPE command
 using ADMIN_CMD procedure 161
 routines
 authorization 2
 built-in
 ADMIN_CMD procedure 4
 administrative 4
 administrative task scheduler 290
 audit 303
 automatic maintenance 306
 configuration 363
 data distribution 1000
 database environment 381
 DB2 Text Search 1013
 environment 381
 explain 402
 miscellaneous 1075
 packages 990
 security 925
 snapshot 938
 SQL 990
 stepwise redistribution 1000
 storage management 1009
 WebSphere MQ routines 904
 workload management 1053
 monitor 459
 SQL
 administrative (deprecated) 1119
 built-in (summary) 4
 routines and views
 administrative routines 4
 administrative task scheduler 290
 audit 303
 automatic maintenance 306
 configuration 363
 data distribution 1000
 database environment 381
 DB2 Text Search 1013
 environment 381
 explain 402
 miscellaneous 1075
 packages 990
 security 925
 snapshot 938

routines and views (continued)

SQL 990
 stepwise redistribution 1000
 storage management 1009
 WebSphere MQ routines 904
 workload management 1053
 RUNSTATS command
 details
 using ADMIN_CMD
 procedure 161

S

scalar functions
 AUTH_GET_INSTANCE_AUTHID 926
 SQLERRM 1115
 schemas
 copying 195
 dropping 199
 objects 195
 security
 routines and views 925
 SET TAPE POSITION command
 using ADMIN_CMD procedure 177
 SET_CONFIG stored procedure 353
 SET_MAINT_MODE_RECORD_NO_TEMPORALHISTORY
 procedure 1114
 SET_ROUTINE_OPTS procedure 999
 SET_SWRD_SETTINGS procedure 1005
 SNAP_GET_AGENT table
 function 1250, 1430
 SNAP_GET_AGENT_MEMORY_POOL
 table function 1254, 1433
 SNAP_GET_APPL table function 1272,
 1446
 SNAP_GET_APPL_INFO table
 function 1258, 1438
 SNAP_GET_APPL_INFO_V95 table
 function 1266
 SNAP_GET_BP table function 1287,
 1455
 SNAP_GET_BP_PART table
 function 1293, 1460
 SNAP_GET_CONTAINER table
 function 1301, 1464
 SNAP_GET_CONTAINER_V91
 deprecated table function 1305
 SNAP_GET_DB table function 1308,
 1468, 1518
 SNAP_GET_DB_MEMORY_POOL table
 function 1320, 1480
 SNAP_GET_DB_V97 table function 1324
 SNAP_GET_DBM table function 1334,
 1484
 SNAP_GET_DBM_MEMORY_POOL table
 function 1339, 1489
 SNAP_GET_DETAILLOG table
 function 1345, 1493
 SNAP_GET_DETAILLOG_V91 table
 function 1349
 SNAP_GET_DYN_SQL table
 function 940, 964
 SNAP_GET_DYN_SQL_V95 table
 function 1351
 SNAP_GET_FCM table function 1355,
 1496

SNAP_GET_FCM_PART table function 1358, 1499
 SNAP_GET_HADR table function 1361, 1503
 SNAP_GET_LOCK table function 1366, 1507
 SNAP_GET_LOCKWAIT table function 1371, 1512
 SNAP_GET_STMT table function 1382, 1534
 SNAP_GET_STO_PATHS deprecated table function 1388
 SNAP_GET_STORAGE_PATHS_V97 table function 1390, 1541
 SNAP_GET_SUBSECTION table function 946, 970
 SNAP_GET_SWITCHES table function 950, 974
 SNAP_GET_TAB table function 1393, 1544
 SNAP_GET_TAB_REORG table function 954, 978
 SNAP_GET_TAB_V91 deprecated table function 1397
 SNAP_GET_TBSP table function 1399, 1548
 SNAP_GET_TBSP_PART table function 1405, 1553
 SNAP_GET_TBSP_PART_V97 table function 1411
 SNAP_GET_TBSP QUIESCER table function 1419, 1559
 SNAP_GET_TBSP_RANGE table function 1377, 1563
 SNAP_GET_TBSP_V91 deprecated table function 1415
 SNAP_GET_UTIL table function 1424, 1530, 1568
 SNAP_GET_UTIL_PROGRESS table function 960, 984
 SNAP_WRITE_FILE procedure 1428
 SNAPAGENT administrative view 1250, 1430
 SNAPAGENT_MEMORY_POOL administrative view 1254, 1433
 SNAPAPPL administrative view 1272, 1446
 SNAPAPPL_INFO administrative view 1258, 1438
 SNAPBP administrative view 1287, 1455
 SNAPBP_PART administrative view 1293, 1460
 SNAPCONTAINER administrative view 1301, 1464
 SNAPDB administrative view 1308, 1468, 1518
 SNAPDB_MEMORY_POOL administrative view 1320, 1480
 SNAPDBM administrative view 1334, 1484
 SNAPDBM_MEMORY_POOL administrative view 1339, 1489
 SNAPDETAILLOG administrative view 1345, 1493
 SNAPDYN_SQL administrative view 940, 964
 SNAPFCM administrative view 1355, 1496
 SNAPFCM_PART administrative view 1358, 1499
 SNAPHADR administrative view 1361, 1503
 SNAPLOCK administrative view 1366, 1507
 SNAPLOCKWAIT administrative view 1371, 1512
 snapshot routines and views 938
 SNAPSTMT administrative view 1382, 1534
 SNAPSTORAGE_PATHS administrative view 1390, 1541
 SNAPSUBSECTION administrative view 946, 970
 SNAPSWITCHES administrative view 950, 974
 SNAPTAB_REORG administrative view 954, 978
 SNAPTbsp administrative view 1399, 1548
 SNAPTbsp_PART administrative view 1405, 1553
 SNAPTbsp QUIESCER administrative view 1419, 1559
 SNAPTbsp_RANGE administrative view 1377, 1563
 SNAPUTIL administrative view 1424, 1530, 1568
 SNAPUTIL_PROGRESS administrative view 960, 984
 split mirrors retrieving database paths 1087
 SQL administrative routines deprecated 1119
 administrative routines and views 990
 SQLCODE returning message information 339
 QLERRM scalar functions 1115
 stepwise redistribution routines and views 1000
 STEPWISE_REDISTRIBUTE_DBPG procedure details 1008
 storage management routines and views 1009
 storage management tool stored procedures 1009, 1011, 1012
 stored procedures
 AUDIT_ARCHIVE 303
 AUDIT_DELIM_EXTRACT 304
 GET_ROUTINE_NAME procedure 991
 MON_CAPTURE_ACTIVITY_IN_PROGRESS 479
 MON_COLLECT_STATS 481
 SET_MAINT_MODE_RECORD_NO_TEMPORAL_HISTORY 114
 WLM_ALTER_MEMBER_SUBSET 1055
 WLM_CREATE_MEMBER_SUBSET 1058
 WLM_DROP_MEMBER_SUBSET 1061
 SYSINSTALLOBJECTS procedure details 1118
 SYSINSTALLROUTINES deprecated procedure 1572
 system information retrieving 346, 401
 SYSTS_ADMIN_CMD procedure 1013
 SYSTS_ALTER stored procedure 1015
 SYSTS_CLEANUP stored procedure 1021
 SYSTS_CLEAR_COMMANDLOCKS stored procedure 1022
 SYSTS_CLEAR_EVENTS stored procedure 1024
 SYSTS_CONFIGURE procedure 1026
 SYSTS_CREATE procedure 1029
 SYSTS_DISABLE procedure 1039
 SYSTS_DROP stored procedure 1041
 SYSTS_ENABLE stored procedure 1043
 SYSTS_UPDATE stored procedure 1045
 SYSTS_UPGRADE_CATALOG stored procedure 1050
 SYSTS_UPGRADE_INDEX stored procedure 1051
T
 table compression information 1126
 table functions
 ADMIN_GET_DBP_MEM_USAGE 1125
 ADMIN_GET_ENCRYPTION_INFO 203
 ADMIN_GET_INDEX_COMPRESS_INFO 205
 ADMIN_GET_INDEX_INFO 209
 ADMIN_GET_MEM_USAGE 214
 ADMIN_GET_MSGS 215
 ADMIN_GET_STORAGE_PATHS 217
 ADMIN_GET_TAB_COMPRESS_INFO 219
 ADMIN_GET_TAB_DICTIONARY_INFO 221
 ADMIN_GET_TAB_INFO 224, 274
 ADMIN_GET_TEMP_COLUMNS 233, 283
 ADMIN_GET_TEMP_TABLES 237, 287
 ADMIN_LIST_DB_PATHS 1087
 AUDIT_ARCHIVE 303
 AUTH_LIST_GROUPS_FOR_AUTHID 931
 built-in views comparison 3
 DB_GET_CFG 363
 DB2_GET_CLUSTER_HOST_STATE details 373
 DB2_GET_INSTANCE_INFO details 376
 deprecated functionality
 ADMIN_GET_TAB_COMPRESS_INFO 1126
 ADMIN_GET_TAB_COMPRESS_INFO_V97 1132
 ADMIN_GET_TAB_INFO_V95 1138
 ADMIN_GET_TAB_INFO_V97 1144
 SNAP_GET_APPL_INFO_V95 1266
 SNAP_GET_BP_V95 1297
 SNAP_GET_DBM_V95 1342
 SNAP_GET_DETAILLOG_V91 1349
 SNAP_GET_DYN_SQL_V95 1351
 summary 1119
 ENV_GET_DB2_EDU_SYSTEM_RESOURCES 384
 ENV_GET_DB2_SYSTEM_RESOURCES 387
 ENV_GET_NETWORK_RESOURCES 388
 ENV_GET_REG_VARIABLES 389

table functions (continued)

ENV_GET_ROLLING_UPDATE_STATUS 392
 ENV_GET_SYSTEM_RESOURCES 395
 HEALTH_GET_ALERT_ACTION_CFG 1217
 HEALTH_GET_ALERT_CFG 1220
 HEALTH_GET_IND_DEFINITION 1223
 MON_FORMAT_LOCK_NAME 483
 MON_GET_ACTIVITY 514
 MON_GET_AGENT 551
 MON_GET_APPL_LOCKWAIT 561
 MON_GET_DATABASE 619
 MON_GET_DATABASE_DETAILS 636
 MON_GET_INSTANCE 668
 MON_GET_LATCH 670
 MON_GET_LOCKS 674
 MON_GET_MEMORY_POOL 679
 MON_GET_MEMORY_SET 682
 MON_GET_QUEUE_STATS 712
 MON_GET_ROUTINE 717
 MON_GET_ROUTINE_DETAILS 733
 MON_GET_ROUTINE_EXEC_LIST 748
 MON_GET_SECTION 754
 MON_GET_SECTION_OBJECT 755
 MON_GET_SECTION_ROUTINE 758
 MON_GET_SERVICE_SUBCLASS_STATS 792
 MON_GET_SERVICE_SUPERCLASS_STATS 798
 MON_GET_TABLESPACE_QUIESCER 817
 MON_GET_TABLESPACE_RANGE 820
 MON_GET_UTILITY 863
 MON_GET_WORK_ACTION_SET_STATS 885
 MON_GET_WORKLOAD_STATS 892
 PD_GET_DIAG_HIST 1094
 PD_GET_LOG_MSGS 1102
 REG_LIST_VARIABLES table
 function 1248
 SNAP_GET_AGENT 1250, 1430
 SNAP_GET_AGENT_MEMORY_POOL 1254,
 1433
 SNAP_GET_APPL 1272, 1446
 SNAP_GET_APPL_INFO 1258, 1438
 SNAP_GET_BP 1287, 1455
 SNAP_GET_BP_PART 1293, 1460
 SNAP_GET_CONTAINER 1301, 1464
 SNAP_GET_DB 1308, 1468, 1518
 SNAP_GET_DB_MEMORY_POOL 1320,
 1480
 SNAP_GET_DB_V97 1324
 SNAP_GET_DBM 1334, 1484
 SNAP_GET_DBM_MEMORY_POOL 1339,
 1489
 SNAP_GET_DETAILLOG 1345, 1493
 SNAP_GET_DETAILLOG_V91 1349
 SNAP_GET_DYN_SQL 940, 964
 SNAP_GET_FCM 1355, 1496
 SNAP_GET_FCM_PART 1358, 1499
 SNAP_GET_HADR 1361, 1503
 SNAP_GET_LOCK 1366, 1507
 SNAP_GET_LOCKWAIT 1371, 1512
 SNAP_GET_STMT 1382, 1534
 SNAP_GET_STORAGE_PATHS_V97 1390,
 1541
 SNAP_GET_SUBSECTION 946, 970
 SNAP_GET_SWITCHES 950, 974
 SNAP_GET_TAB 1393, 1544
 SNAP_GET_TAB_REORG 954, 978
 SNAP_GET_TBSP 1399, 1548
 SNAP_GET_TBSP_PART 1405, 1553

table functions (continued)

SNAP_GET_TBSP_PART_V97 1411
 SNAP_GET_TBSP_QUIESCER 1419,
 1559
 SNAP_GET_TBSP_RANGE 1377,
 1563
 SNAP_GET_UTIL 1424, 1530, 1568
 SNAP_GET_UTIL_PROGRESS 960,
 984
 summary 4
 WLM_GET_SERVICE_CLASS_AGENTS 1064
 WLM_GET_SERVICE_CLASS_WORKLOAD
 _OCCURRENCES 1064
 WLM_GET_SERVICE_SUBCLASS_STATS 1067
 WLM_GET_SERVICE_SUBCLASS_STATS_V97
 1067
 WLM_GET_WORKLOAD_STATS 1069

tables

moving online
 ADMIN_MOVE_TABLE
 procedure 242
 ADMIN_MOVE_TABLE_UTIL
 procedure 266
 retrieving information
 column information for temporary
 tables 233, 283
 size 224, 274, 1144
 state 224, 274, 1144
 temporary tables 237, 287
 TBSP_UTILIZATION administrative
 view 1572
 TOP_DYNAMIC_SQL administrative
 view 988

U

UNQUIESCE DATABASE command
 using ADMIN_CMD procedure 178
 UPDATE ALERT CONFIGURATION
 command
 using ADMIN_CMD procedure 179
 UPDATE CONTACT command
 using ADMIN_CMD procedure 183
 UPDATE CONTACTGROUP command
 using ADMIN_CMD procedure 184
 UPDATE DATABASE CONFIGURATION
 command
 using ADMIN_CMD procedure 185
 UPDATE DATABASE MANAGER
 CONFIGURATION command
 using ADMIN_CMD procedure 189
 UPDATE HEALTH NOTIFICATION
 CONTACT LIST command
 using ADMIN_CMD procedure 191
 UPDATE HISTORY command 191
 UPDATE STMM TUNING
 command 193

V

views

administrative views
 ADMIN_TASK_LIST 296
 ADMIN_TASK_STATUS 299
 ADMINTABCOMPRESSINFO
 (deprecated) 1126
 ADMINTABINFO 224, 274

views (continued)

administrative views (continued)

ADMINTEMPCOLUMNS 233,
 283
 ADMINTEMPTABLES 237, 287
 APPL_PERFORMANCE 1165
 APPLICATIONS 1167
 AUTHORIZATIONIDS 935
 BP_HITRATIO 1172
 BP_READ_IO 1174
 BP_WRITE_IO 1176
 CONTACTGROUPS 1080
 CONTACTS 1082
 CONTAINER_UTILIZATION 1178
 DB1HISTORY 1083
 DB2_CF 376
 DB2_CLUSTER_HOST_STATE 373
 DB2_INSTANCE_ALERTS 375
 DB2_MEMBER 376
 DBCFG 363
 DBMCFG 367
 DBPATHS 1087
 ENV_FEATURE_INFO 383
 ENV_INST_INFO 398
 ENV_PROD_INFO 400
 ENV_SYS_INFO 401
 ENV_SYS_RESOURCES 1182
 LOCKS_HELD 1236
 LOCKWAITS 1239
 LOG_UTILIZATION 1242
 LONG_RUNNING_SQL 1244
 MON_BP_UTILIZATION 422
 MON_CONNECTION_SUMMARY 430
 MON_CURRENT_SQL 435
 MON_CURRENT_UOW 436
 MON_DB_SUMMARY 438
 MON_LOCKWAITS 441
 MON_PKG_CACHE_SUMMARY 443
 MON_SERVICE_SUBCLASS_SUMMARY 445
 MON_TBSP_UTILIZATION 449
 MON_WORKLOAD_SUMMARY 455
 NOTIFICATIONLIST 1093
 OBJECTOWNERS 936
 PDLOGMSG_LAST24HOURS 1102
 PRIVILEGES 937
 QUERY_PREP_COST 938
 REG_VARIABLES 1247
 SNAPAGENT 1250, 1430
 SNAPAGENT_MEMORY_POOL 1254,
 1433
 SNAPAPPL 1272, 1446
 SNAPAPPL_INFO 1258, 1438
 SNAPBP 1287, 1455
 SNAPBP_PART 1293, 1460
 SNAPCONTAINER 1301, 1464
 SNAPDB 1308, 1468, 1518
 SNAPDB_MEMORY_POOL 1320,
 1480
 SNAPDBM 1334, 1484
 SNAPDBM_MEMORY_POOL 1339,
 1489
 SNAPDETAILLOG 1345, 1493
 SNAPDYN_SQL 940, 964
 SNAPFCM 1355, 1496
 SNAPFCM_PART 1358, 1499
 SNAPHADR 1361, 1503
 SNAPLOCK 1366, 1507

views (*continued*)

administrative views (*continued*)

SNAPLOCKWAIT 1371, 1512
SNAPSTMT 1382, 1534
SNAPSTORAGE_PATHS 1390,
1541
SNAPSUBSECTION 946, 970
SNAPSWITCHES 950, 974
SNAPTAB 1393, 1544
SNAPTAB_REORG 954, 978
SNAPTbsp 1399, 1548
SNAPTbsp_PART 1405, 1553
SNAPTbsp_QUIESCER 1419,
1559
SNAPTbsp_RANGE 1377, 1563
SNAPUTIL 1424, 1530, 1568
SNAPUTIL_PROGRESS 960, 984
Tbsp_UTILIZATION 1572
TOP_DYNAMIC_SQL 988
MON_TRANSACTION_LOG_UTILIZATION 454
monitor 421

WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES_V97
table function 1597
WLM_GET_WORKLOAD_STATS table
function 1069
WLM_GET_WORKLOAD_STATS_V97
table function 1601
WLM_SET_CLIENT_INFO
procedure 1070
WLM_SET_CONN_ENV procedure 1072
workload management
routines and views 1053

X

XML
common SQL API input 315
XML documents
output documents
versioning for common SQL
API 313

W

WebSphere MQ routines
routines and views 904
WLM_ALTER_MEMBER_SUBSET stored
procedure 1055
WLM_CANCEL_ACTIVITY
procedure 1053
WLM_CAPTURE_ACTIVITY_IN_PROGRESS
procedure 1054
WLM_COLLECT_STATS procedure
details 1055
WLM_CREATE_MEMBER_SUBSET
stored procedure 1058
WLM_DROP_MEMBER_SUBSET stored
procedure 1061
WLM_GET_ACTIVITY_DETAILS table
function 1575
WLM_GET_CONN_ENV table
function 1061
WLM_GET_QUEUE_STATS table
function 1063
WLM_GET_SERVICE_CLASS_AGENTS
table function
details 1064
WLM_GET_SERVICE_CLASS_AGENTS_V97
table function 1581
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES
table function
details 1064
WLM_GET_SERVICE_CLASS_WORKLOAD_OCCURRENCES
_V97 table function 1588
WLM_GET_SERVICE_SUBCLASS_STATS
table function
details 1067
WLM_GET_SERVICE_SUBCLASS_STATS_V97
table function 1591
WLM_GET_SERVICE_SUPERCLASS_STATS
table function 1068
WLM_GET_WORK_ACTION_SET_STATS
table function
details 1068
WLM_GET_WORKLOAD_OCCURRENCE_ACTIVITIES
table function
details 1069



Printed in USA