

DB2®

IBM

DB2 Version 9
for Linux, UNIX, and Windows



Administrative SQL Routines and Views

DB2®

IBM

DB2 Version 9
for Linux, UNIX, and Windows



Administrative SQL Routines and Views

Before using this information and the product it supports, be sure to read the general information under *Notices*.

Edition Notice

This document contains proprietary information of IBM. It is provided under a license agreement and is protected by copyright law. The information contained in this publication does not include any product warranties, and any statements provided in this manual should not be interpreted as such.

You can order IBM publications online or through your local IBM representative.

- To order publications online, go to the IBM Publications Center at www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide

To order DB2 publications from DB2 Marketing and Sales in the United States or Canada, call 1-800-IBM-4YOU (426-4968).

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2006. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Introduction 1

Administrative SQL routines and views 2

Administrative views versus table functions 3

Chapter 2. Authorization. 5

Authorization for administrative views 6

Chapter 3. Supported administrative SQL routines and views. 7

Supported administrative SQL routines and views. 8

Activity monitor administrative SQL routines and views 18

AM_BASE_RPT_RECOMS – Recommendations for activity reports 18

AM_BASE_RPTS – Activity monitor reports 20

AM_DROP_TASK – Delete a monitoring task 22

AM_GET_LOCK_CHN_TB – Retrieve application lock chain data in a tabular format 23

AM_GET_LOCK_CHNS – Retrieve lock chain information for a specific application 25

AM_GET_LOCK_RPT – Retrieve application lock details 26

AM_GET_RPT – Retrieve activity monitor data 34

AM_SAVE_TASK – Create or modify a monitoring task 36

ADMIN_CMD stored procedure and associated administrative SQL routines 38

ADMIN_CMD – Run administrative commands 38

ADMIN_GET_MSGS table function – Retrieve messages generated by a data movement utility that is executed through the ADMIN_CMD procedure 41

ADMIN_REMOVE_MSGS procedure – Clean up messages generated by a data movement utility that is executed through the ADMIN_CMD procedure 43

ADD CONTACT command using the ADMIN_CMD procedure. 44

ADD CONTACTGROUP command using the ADMIN_CMD procedure. 46

AUTOCONFIGURE command using the ADMIN_CMD procedure. 48

BACKUP DATABASE command using the ADMIN_CMD procedure. 53

DESCRIBE command using the ADMIN_CMD procedure 58

DROP CONTACT command using the ADMIN_CMD procedure. 68

DROP CONTACTGROUP command using the ADMIN_CMD procedure. 69

EXPORT command using the ADMIN_CMD procedure 70

FORCE APPLICATION command using the ADMIN_CMD procedure. 76

GET STMM TUNING DBPARTITIONNUM command using the ADMIN_CMD procedure. 78

IMPORT command using the ADMIN_CMD procedure 80

INITIALIZE TAPE command using the ADMIN_CMD procedure. 94

LOAD command using the ADMIN_CMD procedure 96

PRUNE HISTORY/LOGFILE command using the ADMIN_CMD procedure 115

QUIESCE DATABASE command using the ADMIN_CMD procedure 117

QUIESCE TABLESPACES FOR TABLE command using the ADMIN_CMD procedure 119

REDISTRIBUTE DATABASE PARTITION GROUP command using the ADMIN_CMD procedure 122

REORG INDEXES/TABLE command using the ADMIN_CMD procedure 126

RESET ALERT CONFIGURATION command using the ADMIN_CMD procedure 136

RESET DATABASE CONFIGURATION command using the ADMIN_CMD procedure 139

RESET DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure 141

REWIND TAPE command using the ADMIN_CMD procedure 143

RUNSTATS command using the ADMIN_CMD procedure 144

SET TAPE POSITION command using the ADMIN_CMD procedure 156

UNQUIESCE DATABASE command using the ADMIN_CMD procedure 157

UPDATE ALERT CONFIGURATION command using the ADMIN_CMD procedure 159

UPDATE CONTACT command using the ADMIN_CMD procedure 164

UPDATE CONTACTGROUP command using the ADMIN_CMD procedure 166

UPDATE DATABASE CONFIGURATION command using the ADMIN_CMD procedure 168

UPDATE DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure 171

UPDATE HEALTH NOTIFICATION CONTACT LIST command using the ADMIN_CMD procedure 174

UPDATE HISTORY command using the ADMIN_CMD procedure 176

UPDATE STMM TUNING DBPARTITIONNUM command using the ADMIN_CMD procedure 178

Configuration administrative SQL routines and views 180

DB_PARTITIONS 180

| | | | |
|--|-----|--|-----|
| DBCFCG administrative view – Retrieve database configuration parameter information | 182 | PRIVILEGES administrative view – Retrieve privilege information | 278 |
| DBMCFG administrative view – Retrieve database manager configuration parameter information | 184 | Snapshot administrative SQL routines and views | 280 |
| REG_VARIABLES administrative view – Retrieve DB2 registry settings in use. | 187 | APPLICATIONS administrative view – Retrieve connected database application information | 280 |
| Environment administrative SQL routines and views | 189 | APPL_PERFORMANCE administrative view – Retrieve percentage of rows selected for an application | 286 |
| ENV_INST_INFO administrative view – Retrieve information about the current instance . | 189 | BP_HITRATIO administrative view – Retrieve bufferpool hit ratio information | 288 |
| ENV_PROD_INFO administrative view – Retrieve information about installed DB2 products | 191 | BP_READ_IO administrative view – Retrieve bufferpool read performance information | 290 |
| ENV_SYS_INFO administrative view – Retrieve information about the system | 193 | BP_WRITE_IO administrative view – Retrieve bufferpool write performance information | 292 |
| Health snapshot administrative SQL routines and views | 195 | CONTAINER_UTILIZATION administrative view – Retrieve table space container and utilization information | 294 |
| HEALTH_CONT_HI | 195 | LOCKS_HELD administrative view – Retrieve information on locks held | 297 |
| HEALTH_CONT_HI_HIS | 197 | LOCKWAITS administrative view – Retrieve current lockwaits information | 301 |
| HEALTH_CONT_INFO | 199 | LOG_UTILIZATION administrative view – Retrieve log utilization information | 306 |
| HEALTH_DB_HI | 201 | LONG_RUNNING_SQL administrative view | 308 |
| HEALTH_DB_HI_HIS | 205 | QUERY_PREP_COST administrative view – Retrieve statement prepare time information | 311 |
| HEALTH_DB_HIC | 209 | SNAP_WRITE_FILE procedure | 313 |
| HEALTH_DB_HIC_HIS | 211 | SNAPAGENT administrative view and SNAP_GET_AGENT table function – Retrieve agent logical data group application snapshot information | 315 |
| HEALTH_DB_INFO | 214 | SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function – Retrieve memory_pool logical data group snapshot information | 319 |
| HEALTH_DBM_HI | 216 | SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information | 324 |
| HEALTH_DBM_HI_HIS | 218 | SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information | 334 |
| HEALTH_DBM_INFO | 221 | SNAPBP administrative view and SNAP_GET_BP table function – Retrieve bufferpool logical group snapshot information | 341 |
| HEALTH_GET_ALERT_ACTION_CFG table function – Retrieve health alert action configuration settings. | 223 | SNAPBP_PART administrative view and SNAP_GET_BP_PART table function – Retrieve bufferpool_nodeinfo logical data group snapshot information | 347 |
| HEALTH_GET_ALERT_CFG table function – Retrieve health alert configuration settings | 226 | SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information. | 351 |
| HEALTH_GET_IND_DEFINITION table function – Retrieve health indicator definitions | 230 | SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group | 356 |
| HEALTH_HI_REC | 233 | | |
| HEALTH_TBS_HI | 235 | | |
| HEALTH_TBS_HI_HIS | 238 | | |
| HEALTH_TBS_INFO | 242 | | |
| MQSeries administrative SQL routines and views | 244 | | |
| MQPUBLISH | 244 | | |
| MQREAD | 247 | | |
| MQREADALL | 249 | | |
| MQREADALLCLOB | 252 | | |
| MQREADCLOB | 255 | | |
| MQRECEIVE | 257 | | |
| MQRECEIVEALL | 259 | | |
| MQRECEIVEALLCLOB | 262 | | |
| MQRECEIVECLOB | 265 | | |
| MQSEND | 267 | | |
| MQSUBSCRIBE | 269 | | |
| MQUNSUBSCRIBE | 271 | | |
| Security administrative SQL routines and views | 273 | | |
| AUTH_LIST_GROUPS_FOR_AUTHID table function – Retrieve group membership list for a given authorization ID | 273 | | |
| AUTHORIZATIONIDS administrative view – Retrieve authorization IDs and types | 275 | | |
| OBJECTOWNERS administrative view – Retrieve object ownership information | 276 | | |

| | | | |
|--|-----|--|-----|
| SNAPDB_MEMORY_POOL administrative view and SNAP_GET_DB_MEMORY_POOL table function – Retrieve database level memory usage information | 369 | SNAPTbsp administrative view and SNAP_GET_TBSP_V91 table function – Retrieve tablespace logical data group snapshot information | 441 |
| SNAPDBM administrative view and SNAP_GET_DBM table function – Retrieve the dbm logical grouping snapshot information | 374 | SNAPTbsp_PART administrative view and SNAP_GET_TBSP_PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information | 447 |
| SNAPDBM_MEMORY_POOL administrative view and SNAP_GET_DBM_MEMORY_POOL table function – Retrieve database manager level memory usage information | 379 | SNAPTbsp_QUIESCER administrative view and SNAP_GET_TBSP_QUIESCER table function – Retrieve quiescer table space snapshot information | 452 |
| SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG_V91 table function – Retrieve snapshot information from the detail_log logical data group | 383 | SNAPTbsp_RANGE administrative view and SNAP_GET_TBSP_RANGE table function – Retrieve range snapshot information | 456 |
| SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information | 387 | SNAPUTIL administrative view and SNAP_GET_UTIL table function – Retrieve utility_info logical data group snapshot information | 460 |
| SNAPFCM administrative view and SNAP_GET_FCM table function – Retrieve the fcm logical data group snapshot information | 392 | SNAPUTIL_PROGRESS administrative view and SNAP_GET_UTIL_PROGRESS table function – Retrieve progress logical data group snapshot information | 464 |
| SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function – Retrieve the fcm_node logical data group snapshot information | 395 | TBSP_UTILIZATION administrative view – Retrieve table space configuration and utilization information | 467 |
| SNAPHADR administrative view and SNAP_GET_HADR table function – Retrieve hadr logical data group snapshot information | 398 | TOP_DYNAMIC_SQL administrative view – Retrieve information on the top dynamic SQL statements | 472 |
| SNAPLOCK administrative view and SNAP_GET_LOCK table function – Retrieve lock logical data group snapshot information | 403 | SQL procedure administrative SQL routines and views | 474 |
| SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function – Retrieve lockwait logical data group snapshot information | 409 | GET_ROUTINE_OPTS | 474 |
| SNAPSTMT administrative view and SNAP_GET_STMT table function – Retrieve statement snapshot information | 415 | GET_ROUTINE_SAR | 475 |
| SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS table function – Retrieve automatic storage path information | 421 | PUT_ROUTINE_SAR | 476 |
| SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function – Retrieve subsection logical monitor group snapshot information | 425 | REBIND_ROUTINE_PACKAGE | 478 |
| SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function – Retrieve database snapshot switch state information | 429 | SET_ROUTINE_OPTS | 479 |
| SNAPTAB administrative view and SNAP_GET_TAB_V91 table function – Retrieve table logical data group snapshot information | 432 | Stepwise redistribute administrative SQL routines | 480 |
| SNAPTAB_REORG administrative view and SNAP_GET_TAB_REORG table function – Retrieve table reorganization snapshot information | 436 | ANALYZE_LOG_SPACE procedure – Retrieve log space analysis information | 480 |
| | | GENERATE_DISTFILE procedure – Generate a data distribution file | 483 |
| | | GET_SWRD_SETTINGS procedure – Retrieve redistribute information | 485 |
| | | SET_SWRD_SETTINGS procedure – Create or change redistribute registry | 488 |
| | | STEPWISE_REDISTRIBUTE_DBPG procedure – Redistribute part of database partition group | 491 |
| | | Storage management tool administrative SQL routines | 493 |
| | | CAPTURE_STORAGEMGMT_INFO procedure – Retrieve storage-related information for a given root object | 493 |
| | | CREATE_STORAGEMGMT_TABLES procedure – Create storage management tables | 495 |
| | | DROP_STORAGEMGMT_TABLES procedure – Drop all storage management tables | 497 |
| | | Miscellaneous administrative SQL routines and views | 498 |
| | | ADMIN_COPY_SCHEMA procedure – Copy a specific schema and its objects | 498 |

| | |
|---|-----|
| ADMIN_DROP_SCHEMA procedure – Drop a specific schema and its objects | 503 |
| ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function – Retrieve size and state information for tables | 506 |
| ALTOBJ | 516 |
| APPLICATION_ID | 519 |
| COMPILATION_ENV table function – Retrieve compilation environment elements | 520 |
| CONTACTGROUPS administrative view – Retrieve the list of contact groups | 523 |
| CONTACTS administrative view – Retrieve list of contacts | 525 |
| DB_HISTORY administrative view – Retrieve history file information | 527 |
| DBPATHS administrative view – Retrieve database paths | 532 |
| EXPLAIN_GET_MSGS | 536 |
| GET_DBSIZE_INFO | 539 |
| NOTIFICATIONLIST administrative view – Retrieve contact list for health notification | 542 |
| PDLOGMSG_LAST24HOURS administrative view and PD_GET_LOG_MSGS table function – Retrieve problem determination messages | 543 |
| REORGCHK_IX_STATS procedure – Retrieve index statistics for reorganization evaluation | 550 |
| REORGCHK_TB_STATS procedure – Retrieve table statistics for reorganization evaluation | 553 |
| SQLERRM scalar functions – Retrieves error message information | 555 |
| SYSINSTALLOBJECTS | 558 |

Chapter 4. Deprecated administrative SQL routines 559

| | |
|--|-----|
| Deprecated SQL administrative routines and their replacement routines or views | 559 |
| GET_DB_CONFIG | 563 |
| GET_DBM_CONFIG | 565 |
| SNAP_GET_CONTAINER | 566 |
| SNAP_GET_DB | 568 |
| SNAP_GET_DYN_SQL | 576 |
| SNAP_GET_STO_PATHS | 579 |
| SNAP_GET_TAB | 580 |
| SNAP_GET_TBSP | 582 |
| SNAP_GET_TBSP_PART | 586 |
| SNAPSHOT_AGENT | 589 |
| SNAPSHOT_APPL | 590 |
| SNAPSHOT_APPL_INFO | 596 |

| | |
|------------------------------|-----|
| SNAPSHOT_BP | 599 |
| SNAPSHOT_CONTAINER | 602 |
| SNAPSHOT_DATABASE | 604 |
| SNAPSHOT_DBM | 611 |
| SNAPSHOT_DYN_SQL | 614 |
| SNAPSHOT_FCM | 616 |
| SNAPSHOT_FCMNODE | 618 |
| SNAPSHOT_FILEW | 619 |
| SNAPSHOT_LOCK | 620 |
| SNAPSHOT_LOCKWAIT | 622 |
| SNAPSHOT_QUIESCERS | 624 |
| SNAPSHOT_RANGES | 626 |
| SNAPSHOT_STATEMENT | 628 |
| SNAPSHOT_SUBSECT | 631 |
| SNAPSHOT_SWITCHES | 633 |
| SNAPSHOT_TABLE | 635 |
| SNAPSHOT_TBREORG | 637 |
| SNAPSHOT_TBS | 639 |
| SNAPSHOT_TBS_CFG | 642 |
| SQLCACHE_SNAPSHOT | 645 |
| SYSINSTALLROUTINES | 647 |

Appendix A. DB2 Database technical information 649

| | |
|---|-----|
| Overview of the DB2 technical information | 649 |
| Documentation feedback | 649 |
| DB2 technical library in hardcopy or PDF format | 650 |
| Ordering printed DB2 books | 652 |
| Displaying SQL state help from the command line processor | 653 |
| Accessing different versions of the DB2 Information Center | 654 |
| Displaying topics in your preferred language in the DB2 Information Center | 654 |
| Updating the DB2 Information Center installed on your computer or intranet server | 655 |
| DB2 tutorials | 657 |
| DB2 troubleshooting information | 657 |
| Terms and Conditions | 658 |

Appendix B. Notices 659

| | |
|----------------------|-----|
| Trademarks | 661 |
|----------------------|-----|

Index 663

Contacting IBM 669

Chapter 1. Introduction

Administrative SQL routines and views

The administrative routines and views provide a primary, easy to use programmatic interface to administer DB2[®] through SQL. They encompass a collection of built-in views, table functions, procedures, and scalar functions for performing a variety of DB2 administrative tasks. For example: reorganizing a table, capturing and retrieving monitor data or retrieving the application ID of the current connection.

These routines and views can be invoked from an SQL-based application, a DB2 command line or a command script.

Related reference:

- “Administrative views versus table functions” on page 3
- “Supported administrative SQL routines and views” on page 8
- “C samples” in *Samples Topics*
- “CLI samples” in *Samples Topics*
- “Command Line Processor (CLP) samples” in *Samples Topics*
- “JDBC samples” in *Samples Topics*

Administrative views versus table functions

DB2 Version 9 introduces administrative views that provide an easy-to-use application programming interface to DB2 administrative functions through SQL.

The administrative views fall into three categories:

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

The administrative views are the preferred and only documented interfaces for the views based on catalog views and the views based on table functions with no input parameters because the table functions do not provide any additional information or performance benefits.

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

- The ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO table function: The administrative view retrieves information for all tables in the database. This can have a significant performance impact for large databases. The performance impact can be reduced by using the table function and specifying a schema name, table name, or both as input.
- The PDLOGMSG_LAST24HOURS administrative view and the PD_GET_LOG_MSGS table function: The administrative view, which retrieves notification log messages, provides quick access to data from the previous 24 hours, whereas the table function allows you to retrieve data from a specified period of time.
- All snapshot monitor administrative views and table functions (SNAP* administrative views, SNAP_GET_* table functions): The snapshot monitor administrative views provide access to data from each database partition. The table functions provide the option to choose between data from a single database partition or data aggregated across all database partitions.

Applications that use the table functions instead of the views might need to be changed because the table functions might change from release to release to enable new information to be returned. The new table function will have the same base name as the original function and will be suffixed with '_Vxx' for the version of the product in which it is added (for example, _V91). The administrative views will always be based on the most current version of the table functions, and therefore allow for more application portability. Since the columns might vary from one release to the next, it is recommended that specific columns be selected from the administrative views, or that the result set be described if a **SELECT *** statement is used by an application.

Related reference:

- “Supported administrative SQL routines and views” on page 8

Administrative views versus table functions

Chapter 2. Authorization

Authorization for administrative views

For all administrative views in the SYSIBMADM schema, you need SELECT privilege on the view. This can be validated with the following query to check that your authorization ID, or a group to which you belong, has SELECT privilege (that is, it meets the search criteria and is listed in the GRANTEE column):

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

where <view_name> is the name of the administrative view.

With the exception of SYSIBMADM.AUTHORIZATIONIDS, SYSIBMADM.OBJECTOWNERS, and SYSIBMADM.PRIVILEGES, you also need EXECUTE privilege on the underlying administrative table function. The underlying administrative table function is listed in the authorization section of the administrative view. This can be validated with the following query:

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

where <routine_name> is the name of the underlying administrative table function as listed in the documentation.

Some administrative views require additional authorities beyond SELECT on the view and EXECUTE on the underlying administrative table function. Any additional authority required is documented in the reference information describing the view.

Related reference:

- “Administrative views versus table functions” on page 3
- “Supported administrative SQL routines and views” on page 8

Chapter 3. Supported administrative SQL routines and views

Supported administrative SQL routines and views

The following tables summarize information about the supported administrative SQL routines and views.

- Activity monitor administrative SQL routines: Table 1
- ADMIN_CMD stored procedure and associated administrative SQL routines: Table 2
- Configuration administrative SQL routines and views: Table 3 on page 9
- Environment administrative views: Table 4 on page 9
- Health snapshot administrative SQL routines: Table 5 on page 9
- MQSeries® administrative SQL routines: Table 6 on page 11
- Security administrative SQL routines and views: Table 7 on page 11
- Snapshot administrative SQL routines and views: Table 8 on page 12
- SQL procedures administrative SQL routines: Table 9 on page 15
- Stepwise redistribute administrative SQL routines: Table 10 on page 16
- Storage management tool administrative SQL routines: Table 11 on page 16
- Miscellaneous administrative SQL routines and views: Table 12 on page 16

Table 1. Activity monitor administrative SQL routines

| Routine name | Schema | Description |
|--|---------|--|
| "AM_BASE_RPT_RECOMS – Recommendations for activity reports" on page 18 | SYSPROC | This table function returns recommendations for activity reports used by the activity monitor. |
| "AM_BASE_RPTS – Activity monitor reports" on page 20 | SYSPROC | This table function returns activity reports used by the activity monitor. |
| "AM_DROP_TASK – Delete a monitoring task" on page 22 | SYSPROC | This procedure deletes a monitoring task. |
| "AM_GET_LOCK_CHN_TB – Retrieve application lock chain data in a tabular format" on page 23 | SYSPROC | This procedure returns application lock chain data in tabular format. |
| "AM_GET_LOCK_CHNS – Retrieve lock chain information for a specific application" on page 25 | SYSPROC | This procedure displays lock chains for a specified application using a formatted string. |
| "AM_GET_LOCK_RPT – Retrieve application lock details" on page 26 | SYSPROC | This procedure displays lock details for an application. |
| "AM_GET_RPT – Retrieve activity monitor data" on page 34 | SYSPROC | This procedure displays activity monitor data for a report. |
| "AM_SAVE_TASK – Create or modify a monitoring task" on page 36 | SYSPROC | This procedure creates or modifies a monitoring task. |

Table 2. ADMIN_CMD stored procedure and associated administrative SQL routines

| Routine name | Schema | Description |
|--|---------|---|
| "ADMIN_CMD – Run administrative commands" on page 38 | SYSPROC | This procedure allows you to execute administrative commands (including DB2 command line processor (CLP) commands) by running ADMIN_CMD through a CALL statement. |

Supported administrative SQL routines and views

Table 2. ADMIN_CMD stored procedure and associated administrative SQL routines (continued)

| Routine name | Schema | Description |
|--|---------|--|
| “ADMIN_GET_MSGS table function – Retrieve messages generated by a data movement utility that is executed through the ADMIN_CMD procedure” on page 41 | SYSPROC | This table function is used to retrieve messages generated by data movement utilities that are executed through the ADMIN_CMD procedure. |
| “ADMIN_REMOVE_MSGS procedure – Clean up messages generated by a data movement utility that is executed through the ADMIN_CMD procedure” on page 43 | SYSPROC | This procedure is used to clean up messages generated by data movement utilities that are executed through the ADMIN_CMD procedure. |

Table 3. Configuration administrative SQL routines and views

| Routine or view name | Schema | Description |
|--|-----------|--|
| “DB_PARTITIONS ” on page 180 | SYSPROC | This table function returns the contents of the db2nodes.cfg file in table form. |
| “DBCFCG administrative view – Retrieve database configuration parameter information” on page 182 | SYSIBMADM | This administrative view returns database configuration information. |
| “DBMCFG administrative view – Retrieve database manager configuration parameter information” on page 184 | SYSIBMADM | This administrative view returns database manager configuration information. |
| “REG_VARIABLES administrative view – Retrieve DB2 registry settings in use” on page 187 | SYSIBMADM | This administrative view returns the DB2 registry settings from all database partitions. |

Table 4. Environment administrative views

| View name | Schema | Description |
|---|-----------|--|
| “ENV_INST_INFO administrative view – Retrieve information about the current instance” on page 189 | SYSIBMADM | This administrative view returns information about the current instance. |
| “ENV_PROD_INFO administrative view – Retrieve information about installed DB2 products” on page 191 | SYSIBMADM | This administrative view returns information about installed DB2 products. |
| “ENV_SYS_INFO administrative view – Retrieve information about the system” on page 193 | SYSIBMADM | This administrative view returns information about the system. |

Table 5. Health snapshot administrative SQL routines

| Routine name | Schema | Description |
|-----------------------------------|---------|--|
| “HEALTH_CONT_HI ” on page 195 | SYSPROC | This table function returns a table with health indicator information for containers from a health snapshot of a database. |
| “HEALTH_CONT_HI_HIS ” on page 197 | SYSPROC | This table function returns a table with health indicator history information for containers from a health snapshot of a database. |
| “HEALTH_CONT_INFO ” on page 199 | SYSPROC | This table function returns a table with rolled-up alert state information for containers from a health snapshot of a database. |

Supported administrative SQL routines and views

Table 5. Health snapshot administrative SQL routines (continued)

| Routine name | Schema | Description |
|---|---------|--|
| "HEALTH_DB_HI " on page 201 | SYSPROC | This table function returns a table with health indicator information from a health snapshot of a database. |
| "HEALTH_DB_HI_HIS " on page 205 | SYSPROC | This table function returns a table with health indicator history information from a health snapshot of a database. |
| "HEALTH_DB_HIC " on page 209 | SYSPROC | This table function returns collection health indicator information from a health snapshot of a database. |
| "HEALTH_DB_HIC_HIS " on page 211 | SYSPROC | This table function returns collection health indicator history information from a health snapshot of a database. |
| "HEALTH_DB_INFO " on page 214 | SYSPROC | This table function returns a table with rolled-up alert state information from a health snapshot of one or all databases. |
| "HEALTH_DBM_HI " on page 216 | SYSPROC | This table function returns a table with health indicator information from a health snapshot of the DB2 database manager. |
| "HEALTH_DBM_HI_HIS " on page 218 | SYSPROC | This table function returns a table with health indicator history information from a health snapshot of the DB2 database manager. |
| "HEALTH_DBM_INFO " on page 221 | SYSPROC | This table function returns a table with rolled-up alert state information from a health snapshot of the DB2 database manager. |
| "HEALTH_GET_ALERT_ACTION_CFG table function –Retrieve health alert action configuration settings" on page 223 | SYSPROC | This table function returns health alert action configuration settings for objects of various types (dbm, database, table space, and table space containers) and for various configuration levels (install default, instance, global, and object). |
| "HEALTH_GET_ALERT_CFG table function – Retrieve health alert configuration settings" on page 226 | SYSPROC | This table function returns health alert configuration settings for objects of various types (dbm, database, table space, table space containers) and for various configuration levels (install default, global, and object). |
| "HEALTH_GET_IND_DEFINITION table function – Retrieve health indicator definitions" on page 230 | SYSPROC | This table function returns the health indicator definition. |
| "HEALTH_HI_REC " on page 233 | SYSPROC | This procedure retrieves a set of recommendations that address a health indicator in alert state on a particular DB2 object. |
| "HEALTH_TBS_HI " on page 235 | SYSPROC | This table function returns a table with health indicator information for table spaces from a health snapshot of a database. |
| "HEALTH_TBS_HI_HIS " on page 238 | SYSPROC | This table function returns a table with health indicator history information for table spaces from a health snapshot of a database. |

Supported administrative SQL routines and views

Table 5. Health snapshot administrative SQL routines (continued)

| Routine name | Schema | Description |
|--------------------------------|---------|---|
| "HEALTH_TBS_INFO " on page 242 | SYSPROC | This table function returns a table with rolled-up alert state information for table spaces from a health snapshot of a database. |

Table 6. MQSeries administrative SQL routines

| Routine name | Schema | Description |
|---------------------------------|----------------|--|
| "MQPUBLISH " on page 244 | DB2MQ, DB2MQ1C | This scalar function publishes data to an MQSeries location. |
| "MQREAD " on page 247 | DB2MQ, DB2MQ1C | This scalar function returns a message from an MQSeries location. |
| "MQREADALL " on page 249 | DB2MQ, DB2MQ1C | This table function returns a table with messages and message metadata from an MQSeries location. |
| "MQREADALLCLOB " on page 252 | DB2MQ | This table function returns a table containing messages and message metadata from a specified MQSeries location. |
| "MQREADCLOB " on page 255 | DB2MQ | This scalar function returns a message from a specified MQSeries location. |
| "MQRECEIVE " on page 257 | DB2MQ, DB2MQ1C | This scalar function returns a message from an MQSeries location and removes the message from the associated queue. |
| "MQRECEIVEALL " on page 259 | DB2MQ, DB2MQ1C | This table function returns a table containing the messages and message metadata from an MQSeries location and removes the messages from the associated queue. |
| "MQRECEIVEALLCLOB " on page 262 | DB2MQ | This table function returns a table containing messages and message metadata from a specified MQSeries location. |
| "MQRECEIVECLOB " on page 265 | DB2MQ | This scalar function returns a message from a specified MQSeries location. |
| "MQSEND " on page 267 | DB2MQ, DB2MQ1C | This scalar function sends data to an MQSeries location. |
| "MQSUBSCRIBE " on page 269 | DB2MQ, DB2MQ1C | This scalar function subscribes to MQSeries messages published on a specific topic. |
| "MQUNSUBSCRIBE " on page 271 | DB2MQ, DB2MQ1C | This scalar function unsubscribes from MQSeries messages published on a specific topic. |

Table 7. Security administrative SQL routines and views:

| Routine or view name | Schema | Description |
|--|-----------|--|
| "AUTH_LIST_GROUPS_FOR_AUTHID table function – Retrieve group membership list for a given authorization ID" on page 273 | SYSPROC | This function returns the list of groups of which the given Authorization ID is a member. |
| "AUTHORIZATIONIDS administrative view – Retrieve authorization IDs and types" on page 275 | SYSIBMADM | This administrative view returns a list of authorization IDs that have been granted privileges or authorities, along with their types, for the currently connected database. |

Supported administrative SQL routines and views

Table 7. Security administrative SQL routines and views: (continued)

| Routine or view name | Schema | Description |
|--|-----------|---|
| "OBJECTOWNERS administrative view – Retrieve object ownership information" on page 276 | SYSIBMADM | This administrative view returns all object ownership information for the currently connected database. |
| "PRIVILEGES administrative view – Retrieve privilege information" on page 278 | SYSIBMADM | This administrative view returns all explicit privileges for the currently connected database. |

Table 8. Snapshot administrative SQL routines and views

| Routine or view name | Schema | Description |
|--|--|---|
| "APPLICATIONS administrative view – Retrieve connected database application information" on page 280 | SYSIBMADM | This administrative view returns information on connected database applications. |
| "APPL_PERFORMANCE administrative view – Retrieve percentage of rows selected for an application" on page 286 | SYSIBMADM | This administrative view displays information about the rate of rows selected versus rows read per application. |
| "BP_HITRATIO administrative view – Retrieve bufferpool hit ratio information" on page 288 | SYSIBMADM | This administrative view returns bufferpool hit ratios, including total, data, and index, in the database. |
| "BP_READ_IO administrative view – Retrieve bufferpool read performance information" on page 290 | SYSIBMADM | This administrative view returns bufferpool read performance information. |
| "BP_WRITE_IO administrative view – Retrieve bufferpool write performance information" on page 292 | SYSIBMADM | This administrative view returns bufferpool write performance information per bufferpool. |
| "CONTAINER_UTILIZATION administrative view – Retrieve table space container and utilization information" on page 294 | SYSIBMADM | This administrative view returns information about table space containers and utilization rates. |
| "LOCKS_HELD administrative view – Retrieve information on locks held" on page 297 | SYSIBMADM | This administrative view returns information on current locks held. |
| "LOCKWAITS administrative view – Retrieve current lockwaits information" on page 301 | SYSIBMADM | This administrative view returns information on locks that are waiting to be granted. |
| "LOG_UTILIZATION administrative view – Retrieve log utilization information" on page 306 | SYSIBMADM | This administrative view returns information about log utilization for the currently connected database. |
| "LONG_RUNNING_SQL administrative view" on page 308 | SYSIBMADM | This administrative view returns the longest running SQL statements in the currently connected database. |
| "QUERY_PREP_COST administrative view – Retrieve statement prepare time information" on page 311 | SYSIBMADM | This administrative view returns a list of statements with information about the time required to prepare the statement. |
| "SNAP_WRITE_FILE procedure" on page 313 | SYSPROC | This procedure writes system snapshot data to a file in the tmp subdirectory of the instance directory. |
| "SNAPAGENT administrative view and SNAP_GET_AGENT table function – Retrieve agent logical data group application snapshot information" on page 315 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about agents from an application snapshot, in particular, the agent logical data group. |

Supported administrative SQL routines and views

Table 8. Snapshot administrative SQL routines and views (continued)

| Routine or view name | Schema | Description |
|--|---|---|
| “SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function – Retrieve memory_pool logical data group snapshot information” on page 319 | SYSIBMADM (administrative view), SYSPROC (table function) | This administrative view and table function return information about memory usage at the agent level. |
| “SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information” on page 324 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about applications from an application snapshot, in particular, the appl logical data group. |
| “SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information” on page 334 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about applications from an application snapshot, in particular, the appl_info logical data group. |
| “SNAPBP administrative view and SNAP_GET_BP table function – Retrieve bufferpool logical group snapshot information” on page 341 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about buffer pools from a bufferpool snapshot, in particular, the bufferpool logical data group. |
| “SNAPBP_PART administrative view and SNAP_GET_BP_PART table function – Retrieve bufferpool_nodeinfo logical data group snapshot information” on page 347 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about buffer pools from a bufferpool snapshot, in particular, the bufferpool_nodeinfo logical data group. |
| “SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information” on page 351 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return table space snapshot information from the tablespace_container logical data group. |
| “SNAPDDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group” on page 356 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return snapshot information from the database (dbase) and database storage (db_storage_group) logical groupings. |
| “SNAPDDB_MEMORY_POOL administrative view and SNAP_GET_DB_MEMORY_POOL table function – Retrieve database level memory usage information” on page 369 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about memory usage at the database level for UNIX® platforms only. |
| “SNAPDBM administrative view and SNAP_GET_DBM table function – Retrieve the dbm logical grouping snapshot information” on page 374 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return the snapshot monitor DB2 database manager (dbm) logical grouping information. |
| “SNAPDBM_MEMORY_POOL administrative view and SNAP_GET_DBM_MEMORY_POOL table function – Retrieve database manager level memory usage information” on page 379 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about memory usage at the database manager. |
| “SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG_V91 table function – Retrieve snapshot information from the detail_log logical data group” on page 383 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return snapshot information from the detail_log logical data group. |
| “SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information” on page 387 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return snapshot information from the dynsql logical data group. |

Supported administrative SQL routines and views

Table 8. Snapshot administrative SQL routines and views (continued)

| Routine or view name | Schema | Description |
|--|---|--|
| “SNAPFCM administrative view and SNAP_GET_FCM table function – Retrieve the fcm logical data group snapshot information” on page 392 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about the fast communication manager (FCM) from a database manager snapshot, in particular, the fcm logical data group. |
| “SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function – Retrieve the fcm_node logical data group snapshot information” on page 395 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about the fast communication manager (FCM) from a database manager snapshot, in particular, the fcm_node logical data group. |
| “SNAPHADR administrative view and SNAP_GET_HADR table function – Retrieve hadr logical data group snapshot information” on page 398 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about high availability disaster recovery from a database snapshot, in particular, the hadr logical data group. |
| “SNAPLOCK administrative view and SNAP_GET_LOCK table function – Retrieve lock logical data group snapshot information” on page 403 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return snapshot information about locks, in particular, the lock logical data group. |
| “SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function – Retrieve lockwait logical data group snapshot information” on page 409 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return snapshot information about lock waits, in particular, the lockwait logical data group. |
| “SNAPSTMT administrative view and SNAP_GET_STMT table function – Retrieve statement snapshot information” on page 415 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about statements from an application snapshot. |
| “SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS table function – Retrieve automatic storage path information” on page 421 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function 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 |
| “SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function – Retrieve subsection logical monitor group snapshot information” on page 425 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about application subsections, namely the subsection logical monitor grouping. |
| “SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function – Retrieve database snapshot switch state information” on page 429 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about the database snapshot switch state. |
| “SNAPTAB administrative view and SNAP_GET_TAB_V91 table function – Retrieve table logical data group snapshot information” on page 432 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return snapshot information from the table logical data group. |
| “SNAPTAB_REORG administrative view and SNAP_GET_TAB_REORG table function – Retrieve table reorganization snapshot information” on page 436 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return table reorganization information. |
| “SNAPTbsp administrative view and SNAP_GET_TBSP_V91 table function – Retrieve tablespace logical data group snapshot information” on page 441 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return snapshot information from the tablespace logical data group. |

Supported administrative SQL routines and views

Table 8. Snapshot administrative SQL routines and views (continued)

| Routine or view name | Schema | Description |
|--|---|--|
| "SNAPTbsp_PART administrative view and SNAP_GET_TBSP_PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information" on page 447 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return snapshot information from the tablespace_nodeinfo logical data group. |
| "SNAPTbsp_QUIESCER administrative view and SNAP_GET_TBSP_QUIESCER table function – Retrieve quiescer table space snapshot information" on page 452 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about quiescers from a table space snapshot. |
| "SNAPTbsp_RANGE administrative view and SNAP_GET_TBSP_RANGE table function – Retrieve range snapshot information" on page 456 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information from a range snapshot. |
| "SNAPUTIL administrative view and SNAP_GET_UTIL table function – Retrieve utility_info logical data group snapshot information" on page 460 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return snapshot information on utilities from the utility_info logical data group. |
| "SNAPUTIL_PROGRESS administrative view and SNAP_GET_UTIL_PROGRESS table function – Retrieve progress logical data group snapshot information" on page 464 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return information about utility progress, in particular, the progress logical data group. |
| "TBSP_UTILIZATION administrative view – Retrieve table space configuration and utilization information" on page 467 | SYSIBMADM | This administrative view returns table space configuration and utilization information. |
| "TOP_DYNAMIC_SQL administrative view – Retrieve information on the top dynamic SQL statements" on page 472 | SYSIBMADM | This administrative view returns the top dynamic SQL statements sortable by number of executions, average execution time, number of sorts, or sorts per statement. |

Table 9. SQL procedures administrative SQL routines

| Routine name | Schema | Description |
|---------------------------------------|---------|--|
| "GET_ROUTINE_OPTS " on page 474 | SYSPROC | This scalar 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_SAR " on page 475 | SYSFUN | This procedure returns the information necessary to install an identical routine on another database server running at least at the same level and operating system. |
| "PUT_ROUTINE_SAR " on page 476 | SYSFUN | This procedure passes the information necessary to create and define an SQL routine at the database server. |
| "REBIND_ROUTINE_PACKAGE " on page 478 | SYSPROC | This procedure rebinds the package associated with an SQL procedure. |
| "SET_ROUTINE_OPTS " on page 479 | SYSPROC | This procedure sets the options that are to be used for the creation of SQL procedures in the current session. |

Supported administrative SQL routines and views

Table 10. Stepwise redistribute administrative SQL routines

| Routine name | Schema | Description |
|--|---------|--|
| "ANALYZE_LOG_SPACE procedure – Retrieve log space analysis information" on page 480 | SYSPROC | This procedure returns log space analysis information. |
| "GENERATE_DISTFILE procedure – Generate a data distribution file" on page 483 | SYSPROC | This procedure generates a data distribution file. |
| "GET_SWRD_SETTINGS procedure – Retrieve redistribute information" on page 485 | SYSPROC | This procedure returns redistribute information. |
| "SET_SWRD_SETTINGS procedure – Create or change redistribute registry" on page 488 | SYSPROC | This procedure creates or changes the redistribute registry. |
| "STEPWISE_REDISTRIBUTE_DBPG procedure – Redistribute part of database partition group" on page 491 | SYSPROC | This procedure redistributes part of database partition group. |

Table 11. Storage management tool administrative SQL routines

| Routine name | Schema | Description |
|---|---------|---|
| "CAPTURE_STORAGEMGMT_INFO procedure – Retrieve storage-related information for a given root object" on page 493 | SYSPROC | This procedure returns storage-related information for a given root object. |
| "CREATE_STORAGEMGMT_TABLES procedure – Create storage management tables" on page 495 | SYSPROC | This procedure creates storage management tables. |
| "DROP_STORAGEMGMT_TABLES procedure – Drop all storage management tables" on page 497 | SYSPROC | This procedure drops all storage management tables. |

Table 12. Miscellaneous administrative SQL routines and views

| Routine or view name | Schema | Description |
|---|--|--|
| "ADMIN_COPY_SCHEMA procedure – Copy a specific schema and its objects" on page 498 | SYSPROC | This procedure is used to copy a specific schema and all objects contained in it. |
| "ADMIN_DROP_SCHEMA procedure – Drop a specific schema and its objects" on page 503 | SYSPROC | This procedure is used to drop a specific schema and all objects contained in it. |
| "ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function – Retrieve size and state information for tables" on page 506 | SYSIBMADM (administrative view), SYSPROC (table function) | The administrative view and table function return size and state information for tables, materialized query tables (MQT) and hierarchy tables. |
| "ALTOBJ " on page 516 | SYSPROC | This procedure alters an existing table using the input CREATE TABLE statement as the target table definition. |
| "APPLICATION_ID " on page 519 | SYSFUN | This scalar function returns the application ID of the current connection. |
| "COMPILATION_ENV table function – Retrieve compilation environment elements" on page 520 | SYSPROC | This table function returns the elements of a compilation environment. |

Supported administrative SQL routines and views

Table 12. Miscellaneous administrative SQL routines and views (continued)

| Routine or view name | Schema | Description |
|---|--|---|
| "CONTACTGROUPS administrative view – Retrieve the list of contact groups" on page 523 | SYSIBMADM | This administrative view returns the list of contact groups. |
| "CONTACTS administrative view – Retrieve list of contacts" on page 525 | SYSIBMADM | This administrative view returns the list of contacts defined on the database server. |
| "DB_HISTORY administrative view – Retrieve history file information" on page 527 | SYSIBMADM | This administrative view returns information from the history file that is associated with the currently connected database partition. |
| "DBPATHS administrative view – Retrieve database paths" on page 532 | SYSIBMADM | This administrative view returns the values for database paths required for tasks such as split mirror backups. |
| "EXPLAIN_GET_MSGS " on page 536 | The schema is the same as the Explain table schema. | This table function queries the EXPLAIN_DIAGNOSTIC and EXPLAIN_DIAGNOSTIC_DATA Explain tables, and returns formatted messages. |
| "GET_DBSIZE_INFO " on page 539 | SYSPROC | This procedure calculates the database size and maximum capacity. |
| "NOTIFICATIONLIST administrative view – Retrieve contact list for health notification" on page 542 | SYSIBMADM | This administrative view returns the list of contacts and contact groups that are notified about the health of an instance. |
| "PDLOGMSG_LAST24HOURS administrative view and PD_GET_LOG_MSGS table function – Retrieve problem determination messages" on page 543 | SYSIBMADM (administrative view), SYSPROC (table function) | This administrative view and table function return problem determination log messages that were logged in the DB2 notification log. The information is intended for use by database and system administrators. |
| "REORGCHK_IX_STATS procedure – Retrieve index statistics for reorganization evaluation" on page 550 | SYSPROC | This procedure checks index statistics to determine whether or not there is a need for reorganization. |
| "REORGCHK_TB_STATS procedure – Retrieve table statistics for reorganization evaluation" on page 553 | SYSPROC | This procedure checks table statistics to determine whether or not there is a need for reorganization. |
| "SQLERRM scalar functions – Retrieves error message information" on page 555 | SYSPROC | 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 is a simple interface which takes only an SQLCODE as an input parameter and returns the short message in English. |
| "SYSINSTALLOBJECTS " on page 558 | SYSPROC | This procedure creates or drops the database objects that are required for a specific tool. |

Related concepts:

- "Administrative SQL routines and views" on page 2

Related reference:

- "Deprecated SQL administrative routines and their replacement routines or views" on page 559

Activity monitor administrative SQL routines and views

AM_BASE_RPT_RECOMS – Recommendations for activity reports

The AM_BASE_RPT_RECOMS table function returns recommendations for activity reports used by the activity monitor.

Syntax:

```
►► 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:

EXECUTE privilege on the AM_BASE_RPT_RECOMS table function.

Examples:

Example 1: Request recommendations (in English) for the activity 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 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 13. 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. |
| RECOM_DESCRIPTION | CLOB(32K) | The detailed description of the recommendation. |

Related concepts:

- “Activity Monitor overview” in *System Monitor Guide and Reference*

Related reference:

- “AM_BASE_RPTS – Activity monitor reports” on page 20
- “AM_DROP_TASK – Delete a monitoring task” on page 22
- “AM_GET_LOCK_CHN_TB – Retrieve application lock chain data in a tabular format” on page 23
- “AM_GET_LOCK_CHNS – Retrieve lock chain information for a specific application” on page 25
- “AM_GET_LOCK_RPT – Retrieve application lock details” on page 26
- “AM_GET_RPT – Retrieve activity monitor data” on page 34
- “AM_SAVE_TASK – Create or modify a monitoring task” on page 36
- “Supported administrative SQL routines and views” on page 8

AM_BASE_RPTS – Activity monitor reports

The AM_BASE_RPTS table function returns activity reports used by the activity monitor.

Syntax:

```
►► 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:

EXECUTE privilege on the AM_BASE_RPTS table function.

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 14. Information returned by the AM_BASE_RPTS table function*

| Column name | Data type | Description |
|-------------|----------------|--|
| ID | INTEGER | The unique report ID. |
| 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. |

Related concepts:

- “Activity Monitor overview” in *System Monitor Guide and Reference*

Related reference:

- “AM_GET_LOCK_CHNS – Retrieve lock chain information for a specific application” on page 25
- “AM_BASE_RPT_RECOMS – Recommendations for activity reports” on page 18
- “AM_DROP_TASK – Delete a monitoring task” on page 22
- “AM_GET_LOCK_CHN_TB – Retrieve application lock chain data in a tabular format” on page 23
- “AM_GET_LOCK_RPT – Retrieve application lock details” on page 26
- “AM_GET_RPT – Retrieve activity monitor data” on page 34
- “AM_SAVE_TASK – Create or modify a monitoring task” on page 36
- “Supported administrative SQL routines and views” on page 8

AM_DROP_TASK – Delete a monitoring task

The AM_DROP_TASK procedure deletes a monitoring task. It does not return any data.

Syntax:

▶▶ 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:

EXECUTE privilege on the AM_DROP_TASK procedure.

Example:

Drop the monitoring task with ID 5.

```
CALL SYSPROC.AM_DROP_TASK(5)
```

Related concepts:

- “Activity Monitor overview” in *System Monitor Guide and Reference*

Related reference:

- “AM_BASE_RPTS – Activity monitor reports” on page 20
- “AM_BASE_RPT_RECOMS – Recommendations for activity reports” on page 18
- “AM_GET_LOCK_CHN_TB – Retrieve application lock chain data in a tabular format” on page 23
- “AM_GET_LOCK_CHNS – Retrieve lock chain information for a specific application” on page 25
- “AM_GET_LOCK_RPT – Retrieve application lock details” on page 26
- “AM_GET_RPT – Retrieve activity monitor data” on page 34
- “AM_SAVE_TASK – Create or modify a monitoring task” on page 36
- “Supported administrative SQL routines and views” on page 8

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:

```
▶▶—AM_GET_LOCK_CHN_TB—(—agent-id—)—————▶▶
```

The schema is SYSPROC.

Procedure parameter:

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.

Example:

Retrieve lock chain information for agent ID 68.

```
CALL SYSPROC.AM_GET_LOCK_CHN_TB(68)
```

Information returned:

The procedure returns a table as shown below. Each row of the 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 15. 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 | The agent ID of the application waiting for the lock. |
| APPL_NAME | VARCHAR(255) | The name of the application waiting for the lock. |
| AUTH_ID | VARCHAR(128) | The authorization ID of the application waiting for the lock. |
| APPL_ID | VARCHAR(64) | The application ID of the application waiting for the lock. |

Related concepts:

- “Activity Monitor overview” in *System Monitor Guide and Reference*

Related reference:

AM_GET_LOCK_CHN_TB

- “AM_BASE_RPTS – Activity monitor reports” on page 20
- “AM_BASE_RPT_RECOMS – Recommendations for activity reports” on page 18
- “AM_DROP_TASK – Delete a monitoring task” on page 22
- “AM_GET_LOCK_CHNS – Retrieve lock chain information for a specific application” on page 25
- “AM_GET_LOCK_RPT – Retrieve application lock details” on page 26
- “AM_GET_RPT – Retrieve activity monitor data” on page 34
- “AM_SAVE_TASK – Create or modify a monitoring task” on page 36
- “Supported administrative SQL routines and views” on page 8

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:

```
▶▶—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.

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
```

Related concepts:

- “Activity Monitor overview” in *System Monitor Guide and Reference*

Related reference:

- “AM_BASE_RPTS – Activity monitor reports” on page 20
- “AM_BASE_RPT_RECOMS – Recommendations for activity reports” on page 18
- “AM_DROP_TASK – Delete a monitoring task” on page 22
- “AM_GET_LOCK_CHN_TB – Retrieve application lock chain data in a tabular format” on page 23
- “AM_GET_LOCK_RPT – Retrieve application lock details” on page 26
- “AM_GET_RPT – Retrieve activity monitor data” on page 34
- “AM_SAVE_TASK – Create or modify a monitoring task” on page 36
- “Supported administrative SQL routines and views” on page 8

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:

```
▶▶ 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.

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 16. 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) monitor element |
| APPL_NAME | VARCHAR(256) | appl_name - Application Name monitor element |
| AUTH_ID | VARCHAR(128) | auth_id - Authorization ID monitor element |
| APPL_ID | VARCHAR(128) | appl_id - Application ID monitor element |

Table 16. General application information returned by the AM_GET_LOCK_RPT procedure (continued)

| Column name | Data Type | Description |
|---------------------|--------------|--|
| APPL_STATUS | VARCHAR(22) | <p>appl_status - Application Status monitor element. 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 |
| COORD_PARTITION_NUM | SMALLINT | coord_node - Coordinating Node monitor element |
| SEQUENCE_NO | VARCHAR(4) | sequence_no - Sequence Number monitor element |
| CLIENT_PRDID | VARCHAR(128) | client_prdid - Client Product/Version ID monitor element |
| CLIENT_PID | BIGINT | client_pid - Client Process ID monitor element |

AM_GET_LOCK_RPT

Table 16. General application information returned by the AM_GET_LOCK_RPT procedure (continued)

| Column name | Data Type | Description |
|-----------------|-------------|---|
| CLIENT_PLATFORM | VARCHAR(12) | client_platform - Client Operating Platform monitor element. This interface returns a text identifier based on the defines in sqlmon.h. <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 16. 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 monitor element. This interface returns a text identifier based on the defines in sqlmon.h, <ul style="list-style-type: none"> • APPC • APPN • CPIC • IPXSPX • LOCAL • NETBIOS • NPIPE • TCPIP (for DB2 Universal Database™, or DB2 UDB) • TCPIP4 • TCPIP6 |
| CLIENT_NNAME | VARCHAR(128) | client_nname - Configuration NNAME of Client monitor element |
| LOCKS_HELD | BIGINT | locks_held - Locks Held monitor element |
| LOCK_WAIT_START_TIME | TIMESTAMP | lock_wait_start_time - Lock Wait Start Timestamp monitor element |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time Waited On Locks monitor element |
| LOCK_WAITS | BIGINT | lock_waits - Lock Waits monitor element |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of Lock Timeouts monitor element |
| LOCK_ESCALS | BIGINT | lock_escalations - Number of Lock Escalations monitor element |
| X_LOCK_ESCALS | BIGINT | x_lock_escalations - Exclusive Lock Escalations monitor element |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks Detected monitor element |

Table 17. 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 monitor element |
| TABSHEMA | VARCHAR(128) | table_schema - Table Schema Name monitor element |
| TABNAME | VARCHAR(128) | table_name - Table Name monitor element |

AM_GET_LOCK_RPT

Table 17. 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 monitor element. 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 monitor element. 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 • NX • S • SIX • U • W • X • Z |

Table 17. 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 monitor element. 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 monitor element |
| LOCK_NAME | VARCHAR(32) | lock_name - Lock Name monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Table 18. 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 monitor element |
| APPL_ID_HOLDING_LK | VARCHAR(128) | appl_id_holding_lk - Application ID Holding Lock monitor element |
| LOCK_WAIT_START_TIME | TIMESTAMP | lock_wait_start_time - Lock Wait Start Timestamp monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |
| TABSCHEMA | VARCHAR(128) | table_schema - Table Schema Name monitor element |
| TABNAME | VARCHAR(128) | table_name - Table Name monitor element |

AM_GET_LOCK_RPT

Table 18. 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 monitor element. 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 monitor element. 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 • NX • S • SIX • U • W • X • Z |

Table 18. 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 monitor element. 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 • NX • S • SIX • U • W • X • Z |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock Escalation monitor element |

Related concepts:

- “Activity Monitor overview” in *System Monitor Guide and Reference*

Related reference:

- “AM_BASE_RPT_RECOMS – Recommendations for activity reports” on page 18
- “AM_BASE_RPTS – Activity monitor reports” on page 20
- “AM_DROP_TASK – Delete a monitoring task” on page 22
- “AM_GET_LOCK_CHN_TB – Retrieve application lock chain data in a tabular format” on page 23
- “AM_GET_LOCK_CHNS – Retrieve lock chain information for a specific application” on page 25
- “AM_GET_RPT – Retrieve activity monitor data” on page 34
- “AM_SAVE_TASK – Create or modify a monitoring task” on page 36
- “Supported administrative SQL routines and views” on page 8
- “dft_monswitches - Default database system monitor switches configuration parameter” in *Performance Guide*

AM_GET_RPT – Retrieve activity monitor data

The AM_GET_RPT procedure returns activity monitor data for a report.

Syntax:

```
▶▶ AM_GET_RPT(—database partition—,—report-id—,—appl-filter—,——————▶
▶—max-number—)—————▶▶▶
```

The schema is SYSPROC.

Procedure parameters:

database 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 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.

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.

Example:

```
CALL SYSPROC.AM_GET_RPT(-2, 18,
  CAST('AGENT_ID=29 AND 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 Activity Monitor graphical tool. To build reports that can be parsed, snapshot administrative SQL routines and views should be used instead.

Related concepts:

- “Activity Monitor overview” in *System Monitor Guide and Reference*

Related reference:

- “AM_BASE_RPTS – Activity monitor reports” on page 20
- “AM_BASE_RPT_RECOMS – Recommendations for activity reports” on page 18

- “AM_DROP_TASK – Delete a monitoring task” on page 22
- “AM_GET_LOCK_CHNS – Retrieve lock chain information for a specific application” on page 25
- “AM_GET_LOCK_RPT – Retrieve application lock details” on page 26
- “AM_GET_LOCK_CHN_TB – Retrieve application lock chain data in a tabular format” on page 23
- “AM_SAVE_TASK – Create or modify a monitoring task” on page 36
- “Supported administrative SQL routines and views” on page 8

AM_SAVE_TASK – Create or modify a monitoring task

The AM_SAVE_TASK procedure creates or modifies a monitoring task.

Syntax:

```
▶▶ 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:

EXECUTE privilege on the AM_SAVE_TASK procedure.

Example:

Example:

```
CALL SYSPROC.AM_SAVE_TASK('M',11,'Task ABC',CAST (NULL AS CLOB(32K)),
    'N','1,2,4,8,9,12')
```

Related concepts:

- “Activity Monitor overview” in *System Monitor Guide and Reference*

Related reference:

- “AM_BASE_RPTS – Activity monitor reports” on page 20
- “AM_BASE_RPT_RECOMS – Recommendations for activity reports” on page 18
- “AM_DROP_TASK – Delete a monitoring task” on page 22
- “AM_GET_LOCK_CHNS – Retrieve lock chain information for a specific application” on page 25
- “AM_GET_LOCK_RPT – Retrieve application lock details” on page 26
- “AM_GET_RPT – Retrieve activity monitor data” on page 34
- “AM_GET_LOCK_CHN_TB – Retrieve application lock chain data in a tabular format” on page 23
- “Supported administrative SQL routines and views” on page 8

ADMIN_CMD stored procedure and associated administrative SQL routines

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:

EXECUTE privilege on the ADMIN_CMD procedure.

The procedure currently supports the following DB2 command line processor (CLP) commands:

- “ADD CONTACT command using the ADMIN_CMD procedure” on page 44
- “ADD CONTACTGROUP command using the ADMIN_CMD procedure” on page 46
- “AUTOCONFIGURE command using the ADMIN_CMD procedure” on page 48
- “BACKUP DATABASE command using the ADMIN_CMD procedure” on page 53 - online only
- “DESCRIBE command using the ADMIN_CMD procedure” on page 58
- “DROP CONTACT command using the ADMIN_CMD procedure” on page 68
- “DROP CONTACTGROUP command using the ADMIN_CMD procedure” on page 69
- “EXPORT command using the ADMIN_CMD procedure” on page 70
- “FORCE APPLICATION command using the ADMIN_CMD procedure” on page 76
- “IMPORT command using the ADMIN_CMD procedure” on page 80
- “INITIALIZE TAPE command using the ADMIN_CMD procedure” on page 94
- “LOAD command using the ADMIN_CMD procedure” on page 96
- “PRUNE HISTORY/LOGFILE command using the ADMIN_CMD procedure” on page 115
- “QUIESCE DATABASE command using the ADMIN_CMD procedure” on page 117
- “QUIESCE TABLESPACES FOR TABLE command using the ADMIN_CMD procedure” on page 119
- “REDISTRIBUTE DATABASE PARTITION GROUP command using the ADMIN_CMD procedure” on page 122

- “REORG INDEXES/TABLE command using the ADMIN_CMD procedure” on page 126
- “RESET ALERT CONFIGURATION command using the ADMIN_CMD procedure” on page 136
- “RESET DATABASE CONFIGURATION command using the ADMIN_CMD procedure” on page 139
- “RESET DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure” on page 141
- “REWIND TAPE command using the ADMIN_CMD procedure” on page 143
- “RUNSTATS command using the ADMIN_CMD procedure” on page 144
- “SET TAPE POSITION command using the ADMIN_CMD procedure” on page 156
- “UNQUIESCE DATABASE command using the ADMIN_CMD procedure” on page 157
- “UPDATE ALERT CONFIGURATION command using the ADMIN_CMD procedure” on page 159
- “UPDATE CONTACT command using the ADMIN_CMD procedure” on page 164
- “UPDATE CONTACTGROUP command using the ADMIN_CMD procedure” on page 166
- “UPDATE DATABASE CONFIGURATION command using the ADMIN_CMD procedure” on page 168
- “UPDATE DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure” on page 171
- “UPDATE HEALTH NOTIFICATION CONTACT LIST command using the ADMIN_CMD procedure” on page 174
- “UPDATE HISTORY command using the ADMIN_CMD procedure” on page 176

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 DBPARTITIONNUM command using the ADMIN_CMD procedure” on page 78
- “UPDATE STMM TUNING DBPARTITIONNUM command using the ADMIN_CMD procedure” on page 178

Usage notes:

Retrieving command execution information:

- Since the ADMIN_CMD procedure runs on the server, the utility messages are 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 19. 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 DB2 CLI applications, but not from embedded C applications.

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.

Related reference:

- “Supported administrative SQL routines and views” on page 8

Related samples:

- “spclient.c -- Call various stored procedures”
- “SpClient.java -- Call a variety of types of stored procedures from SpServer.java (JDBC)”

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_MSGSPATH. 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.

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 is an example of 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.

ADMIN_GET_MSGS

Information returned:

Table 20. Information returned by the ADMIN_GET_MSGS table function

| Column name | Data type | Description |
|----------------|---------------|--|
| DBPARTITIONNUM | INTEGER | Database partition number. This value is only returned for a distributed load and indicates which database partition the corresponding message is for. |
| AGENTTYPE | CHAR(4) | Agent type. This value is only returned for a distributed load. The possible values are: <ul style="list-style-type: none">• 'LOAD': for load agent• 'PART': for partitioning agent• 'PREP': for pre-partitioning agent• NULL: no agent type information is available |
| SQLCODE | VARCHAR(9) | SQLCODE of the message being returned. |
| MSG | VARCHAR(1024) | Short error message that corresponds to the SQLCODE. |

Related reference:

- "Supported administrative SQL routines and views" on page 8
- "Miscellaneous variables" in *Performance Guide*
- "ADMIN_CMD – Run administrative commands" on page 38

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.

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.

Related reference:

- “Miscellaneous variables” in *Performance Guide*
- “Supported administrative SQL routines and views” on page 8
- “ADMIN_CMD – Run administrative commands” on page 38

ADD CONTACT command using the ADMIN_CMD procedure

The command 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.

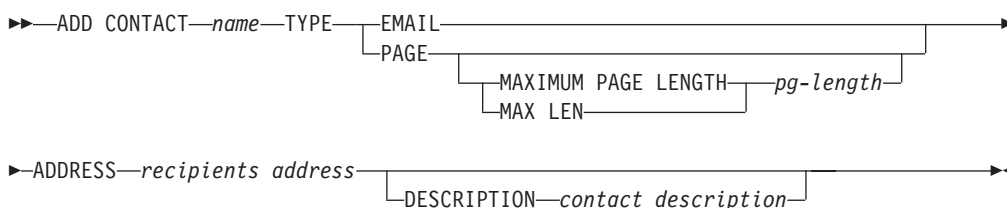
Authorization:

None.

Required connection:

Database. The DAS must be running.

Command syntax:



Command parameters:

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 wishes to be notified by e-mail at (ADDRESS).

PAGE

This contact wishes 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 e-mail 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.

Example:

ADD CONTACT using ADMIN_CMD

Add a contact for user 'testuser' with e-mail address 'testuser@test.com'.

```
CALL SYSPROC.ADMIN_CMD  
('add contact testuser type email address testuser@test.com')
```

Usage notes:

The DAS must have been created and be running.

Command execution status is returned in the SQLCA resulting from the CALL statement.

Related tasks:

- "Enabling health alert notification" in *System Monitor Guide and Reference*

Related reference:

- "ADMIN_CMD – Run administrative commands" on page 38
- "ADD CONTACTGROUP command using the ADMIN_CMD procedure" on page 46
- "DROP CONTACT command using the ADMIN_CMD procedure" on page 68
- "DROP CONTACTGROUP command using the ADMIN_CMD procedure" on page 69
- "UPDATE CONTACT command using the ADMIN_CMD procedure" on page 164
- "UPDATE CONTACTGROUP command using the ADMIN_CMD procedure" on page 166
- "db2admin - DB2 administration server command" in *Command Reference*
- "db2AddContactGroup API - Add a contact group to whom notification messages can be sent" in *Administrative API Reference*

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.

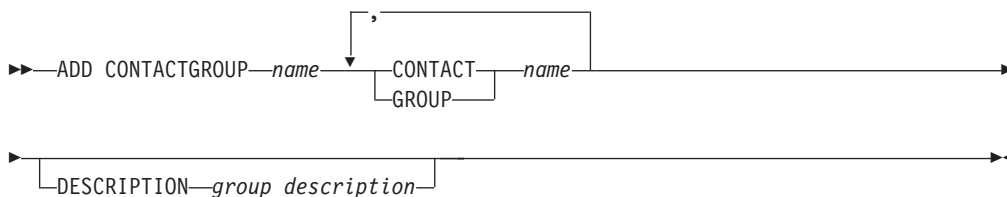
Authorization:

None

Required connection:

Database. The DAS must be running.

Command Syntax:



Command Parameters:

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.

Example:

Create a contact group named 'gname1' that contains two contacts: 'cname1' and 'cname2'.

```
CALL SYSPROC.ADMIN_CMD( 'add contactgroup gname1 contact cname1, contact cname2' )
```

Usage notes:

The DAS must have been created and be running.

Command execution status is returned in the SQLCA resulting from the CALL statement.

ADD CONTACTGROUP using ADMIN_CMD

Related tasks:

- “Enabling health alert notification” in *System Monitor Guide and Reference*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “ADD CONTACT command using the ADMIN_CMD procedure” on page 44
- “DROP CONTACT command using the ADMIN_CMD procedure” on page 68
- “DROP CONTACTGROUP command using the ADMIN_CMD procedure” on page 69
- “UPDATE CONTACT command using the ADMIN_CMD procedure” on page 164
- “UPDATE CONTACTGROUP command using the ADMIN_CMD procedure” on page 166
- “db2admin - DB2 administration server command” in *Command Reference*
- “db2AddContactGroup API - Add a contact group to whom notification messages can be sent” in *Administrative API Reference*

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 recommended values.

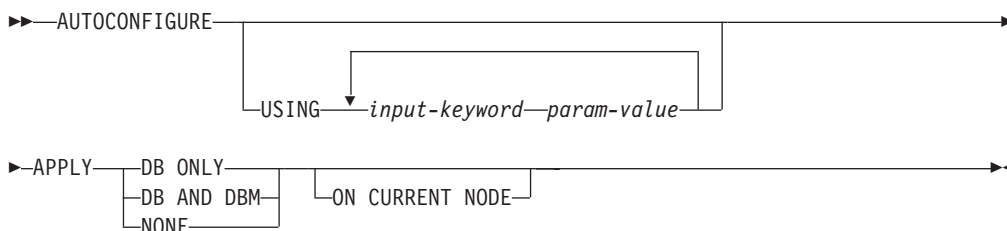
Authorization:

sysadm.

Required connection:

Database.

Command syntax:



Command parameters:

USING input-keyword param-value

Table 21. Valid input keywords and parameter values

| Keyword | Valid values | Default value | Explanation |
|---------------|------------------------|---------------|--|
| mem_percent | 1–100 | 25 | Percentage of memory to dedicate. If other applications (other than the operating system) are running on this server, set this to less than 100. |
| 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 |

Table 21. Valid input keywords and parameter values (continued)

| Keyword | Valid values | Default value | Explanation |
|-----------------|-----------------------------|----------------------|--|
| 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 |
| 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? |

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 NODE

In the Database Partitioning Feature (DPF), the Configuration Advisor updates the database configuration on all nodes by default. Running with the "ON CURRENT NODE" option makes the advisor apply the recommended database configuration to the coordinator (connection) node only.

The bufferpool changes are always applied to the system catalogs. Thus, all nodes are affected. The "ON CURRENT NODE" option does not matter for bufferpool recommendations.

AUTOCONFIGURE using ADMIN_CMD

Example:

Invoke autoconfigure on a database through the ADMIN_CMD stored procedure.
CALL SYSPROC.ADMIN_CMD('AUTOCONFIGURE APPLY NONE')

The following is an example of the result set returned by the command.

| LEVEL | NAME | VALUE | RECOMMENDED_VALUE | DATATYPE |
|-------|-----------------|-------|-------------------|----------|
| DBM | ASLHEAPSZ | 15 | 15 | BIGINT |
| DBM | FCM_NUM_BUFFERS | 512 | 512 | BIGINT |
| ... | | | | |
| DB | APP_CTL_HEAP_SZ | 128 | 144 | INTEGER |
| DB | APPGROUP_MEM_SZ | 20000 | 14559 | BIGINT |
| ... | | | | |
| BP | IBMDEFAULTBP | 1000 | 164182 | BIGINT |

Usage notes:

- On systems with multiple logical partitions, the *mem_percent* parameter refers to the percentage of memory that is to be used by all logical partitions. For example, if DB2 uses 25% of the memory on the system, specify 25% regardless of the number of logical partitions. The database configuration recommendations made, however, will be adjusted for one logical partition.
- This command makes configuration recommendations for the currently connected database, assuming that the database is the only active database on the system. If more than one database is active on the system, adjust the *>mem_percent* parameter to reflect the current database's share of memory. For example, if the DB2 database uses 80% of the system's memory and there are two active databases on the system that should share the resources equally, specify 40% (80% divided by 2 databases) for the parameter *mem_percent*.
- 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.
- Command execution status is returned in the SQLCA resulting from the CALL statement.
- SQL executed in the ADMIN_CMD procedure on behalf of AUTOCONFIGURE is monitored by Query Patroller.
- The AUTOCONFIGURE command issues a COMMIT statement at the end of its execution. In the case of Type-2 connections this will cause the ADMIN_CMD procedure to return SQL30090N with reason code 2.

Result set information:

Command execution status is returned in the SQLCA resulting from the CALL statement. If execution is successful, the command returns additional information the following result set:

Table 22. Result set returned by the AUTOCONFIGURE command

| Column name | Data type | Description |
|-------------------|--------------|---|
| LEVEL | VARCHAR(3) | Level of parameter and is one of: <ul style="list-style-type: none"> • BP for bufferpool level • DBM for database manager level • DB for database level |
| NAME | VARCHAR(128) | <ul style="list-style-type: none"> • If LEVEL is DB or DBM, this contains the configuration parameter keyword. • If LEVEL is BP, this value contains the buffer pool name. |
| VALUE | VARCHAR(256) | <ul style="list-style-type: none"> • If LEVEL is DB or DBM, and the recommended values were applied, this column contains the value of the configuration parameter identified in the NAME column prior to applying the recommended value (that is, it contains the old value). If the change was not applied, this column contains the current on-disk (deferred value) of the identified configuration parameter. • If LEVEL is BP, and the recommended values were applied, this column contains the size (in pages) of the bufferpool identified in the NAME column prior to applying the recommended value (that is, it contains the old size). If the change was not applied, this column contains the current size (in pages) of the identified bufferpool. |
| RECOMMENDED_VALUE | VARCHAR(256) | <ul style="list-style-type: none"> • If LEVEL is DB or DBM, this column contains the recommended (or applied) value of the configuration parameter identified in the parameter column. • If type is BP, this column contains the recommended (or applied) size (in pages) of the bufferpool identified in the parameter column. |
| DATATYPE | VARCHAR(128) | Parameter data type. |

Related concepts:

- “Configuration parameters” in *Performance Guide*

Related tasks:

AUTOCONFIGURE using ADMIN_CMD

- “Defining the scope of configuration parameters using the Configuration Advisor” in *Administration Guide: Implementation*
- “Configuring DB2 with configuration parameters” in *Performance Guide*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “db2AutoConfig API - Access the Configuration Advisor” in *Administrative API Reference*
- “UPDATE DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure” on page 171

BACKUP DATABASE command using the ADMIN_CMD procedure

Creates a backup copy of a database or a table space.

For information on the backup operations supported by DB2 database systems between different operating systems and hardware platforms, see "Backup and restore operations between different operating systems and hardware platforms" in the *Related concepts* section.

Scope:

This command only affects the database partition on which it is executed.

Authorization:

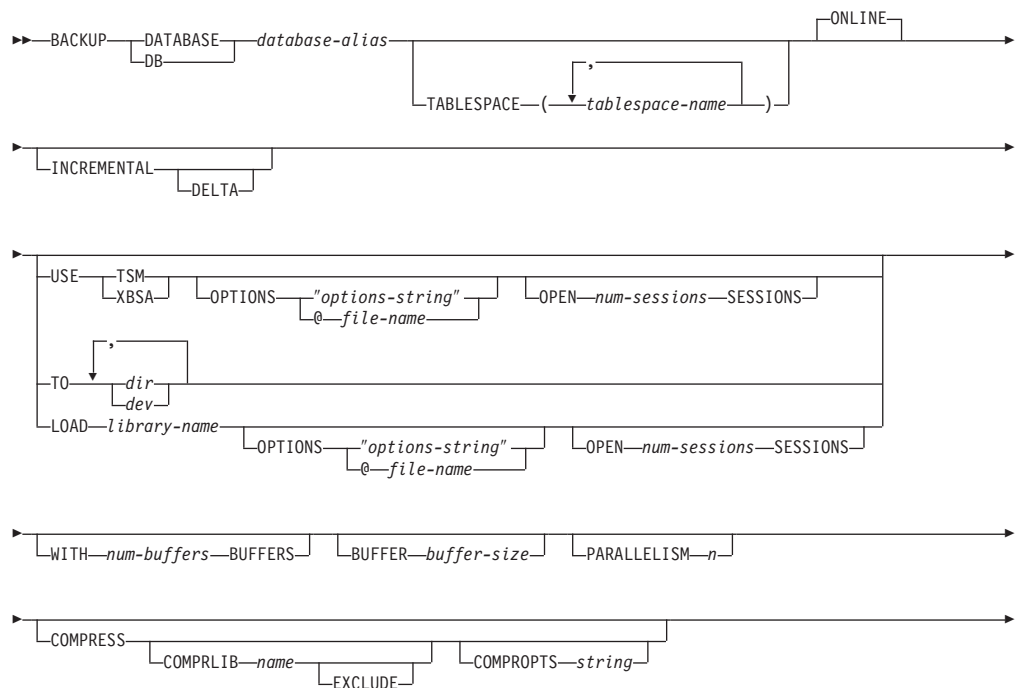
One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

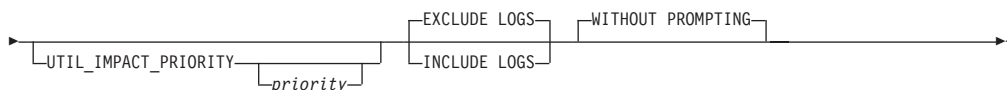
Required connection:

Database. The existing database connection remains after the completion of the backup operation.

Command syntax:



BACKUP DATABASE using ADMIN_CMD



Command parameters:

DATABASE database-alias

Specifies the alias of the database to back up. The alias must be a local database defined on the server and must be the database name that the user is currently connected to. If the database-alias is not the one the user is connected to, an SQL20322N error is returned.

TABLESPACE tablespace-name

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

ONLINE

Specifies online backup. This is the only supported mode and is the default. The ONLINE clause does not need to be specified.

INCREMENTAL

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

DELTA

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

USE TSM

Specifies that the backup is to use Tivoli Storage Manager output.

USE XBSA

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

OPTIONS

"options-string"

Specifies options to be used for the backup operation. The string will be passed to the vendor support library, for example TSM, exactly as it was entered, without the quotes. Specifying this option overrides the value specified by the VENDOROPT database configuration parameter.

@file-name

Specifies that the options to be used for the backup operation are contained in a file located on the DB2 server. The string will be passed to the vendor support library, for example TSM. The file must be a fully qualified file name.

OPEN num-sessions SESSIONS

The number of I/O sessions to be created between DB2 and TSM or another backup vendor product. This parameter has no effect when backing up to tape, disk, or other local device.

TO dir/dev

A list of directory or tape device names. The full path on which the directory resides must be specified. This target directory or device must exist on the database server. This parameter can be repeated to specify the

target directories and devices that the backup image will span. If more than one target is specified (target1, target2, and target3, for example), target1 will be 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. To back up to the FAT file system on Windows operating systems, users must conform to the 8.3 naming restriction.

Use of tape devices or floppy disks might require prompts and user interaction, which will result an error being returned.

- 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, it is recommended that multiple backup copies of the same database 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 will default to the path on which the user exit program resides.

WITH num-buffers BUFFERS

The number of buffers to be used. DB2 will automatically choose an optimal value for this parameter unless you explicitly enter a value. However, when creating a backup to multiple locations, a larger number of buffers can be used to improve performance.

BUFFER buffer-size

The size, in 4 KB pages, of the buffer used when building the backup image. DB2 will automatically choose an optimal value for this parameter unless you explicitly enter a value. The minimum value for this parameter is 8 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 DB2's default buffer size 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:

$$\text{bufferpages} \leq \text{ST_MAX_BUFFERS} * \text{ST_BUFFER_BLOCKS} / 4$$

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 which can be read in parallel by the backup utility. DB2 will automatically choose an optimal value for this parameter unless you explicitly enter a value.

BACKUP DATABASE using ADMIN_CMD

UTIL_IMPACT_PRIORITY *priority*

Specifies that the backup will run 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 will run with the default priority of 50. If UTIL_IMPACT_PRIORITY is not specified, the backup will run 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

Indicates that the backup is to be compressed.

COMPRLIB *name*

Indicates the name of the library to be used to perform the compression. The name must be a fully qualified path referring to a file on the server. If this parameter is not specified, the default DB2 compression library will be used. If the specified library cannot be loaded, the backup will fail.

EXCLUDE

Indicates that the compression library will not be stored in the backup image.

COMPROPTS *string*

Describes a block of binary data that will be passed to the initialization routine in the compression library. DB2 will pass this string directly from the client to the server, so any issues of byte reversal or code page conversion will have to be handled by the compression library. If the first character of the data block is '@', the remainder of the data will be interpreted by DB2 as the name of a file residing on the server. DB2 will then replace the contents of string with the contents of this file and will pass this new value to the initialization routine instead. The maximum length for *string* is 1024 bytes.

EXCLUDE LOGS

Specifies that the backup image should not include any log files. When performing an offline backup operation, logs are excluded whether or not this option is specified.

INCLUDE LOGS

Specifies that the backup image should include the range of log files required to restore and roll forward this image to some consistent point in time. This option is not valid for an offline backup.

WITHOUT PROMPTING

Specifies that the backup will run unattended, and that any actions which normally require user intervention will return an error message. This is the default.

Examples:

The following is a sample weekly incremental backup strategy for a recoverable database. It includes a weekly full database backup operation, a daily non-cumulative (delta) backup operation, and a mid-week cumulative (incremental) backup operation:

BACKUP DATABASE using ADMIN_CMD

```
(Sun) CALL SYSPROC.ADMIN_CMD('backup db sample online use tsm')
(Mon) CALL SYSPROC.ADMIN_CMD
      ('backup db sample online incremental delta use tsm')
(Tue) CALL SYSPROC.ADMIN_CMD
      ('backup db sample online incremental delta use tsm')
(Wed) CALL SYSPROC.ADMIN_CMD
      ('backup db sample online incremental use tsm')
(Thu) CALL SYSPROC.ADMIN_CMD
      ('backup db sample online incremental delta use tsm')
(Fri) CALL SYSPROC.ADMIN_CMD
      ('backup db sample online incremental delta use tsm')
(Sat) CALL SYSPROC.ADMIN_CMD
      ('backup db sample online incremental use tsm')
```

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 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.

Result set information:

Command execution status is returned in the SQLCA resulting from the CALL statement. If execution is successful, and the command returns additional information the following result set:

Table 23. Result set returned by the BACKUP command

| Column name | Data type | Description |
|-------------|-------------|--|
| BACKUP_TIME | VARCHAR(14) | Corresponds to the timestamp string used to name the backup image. |

Related concepts:

- “Developing a backup and recovery strategy” in *Data Recovery and High Availability Guide and Reference*

Related tasks:

- “Using backup” in *Data Recovery and High Availability Guide and Reference*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “db2Backup API - Back up a database or table space” in *Administrative API Reference*

DESCRIBE command using the ADMIN_CMD procedure

This command:

- Displays the output SQLDA information about a SELECT, CALL, or XQuery statement
- Displays columns of a table or a view
- Displays indexes of a table or a view
- Displays data partitions of a table or view

Authorization:

To display the output SQLDA information about a SELECT statement, one of the privileges or authorities listed below for each table or view referenced in the SELECT statement is required.

To display the columns, indexes or data partitions of a table or a view, SELECT privilege, CONTROL privilege, *sysadm* authority or *dbadm* authority is required for the following system catalogs:

- SYSCAT.COLUMNS (DESCRIBE TABLE), SYSCAT.DATAPARTITIONEXPRESSION (with SHOW DETAIL)
- SYSCAT.INDEXES (DESCRIBE INDEXES FOR TABLE) execute privilege on GET_INDEX_COLNAMES() UDF (with SHOW DETAIL)
- SYSCAT.DATAPARTITIONS (DESCRIBE DATA PARTITIONS FOR TABLE)

As PUBLIC has all the privileges over declared global temporary tables, a user can use the command to display information about any declared global temporary table that exists within its connection.

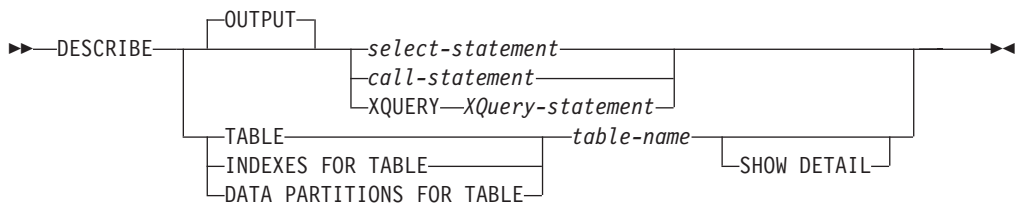
To display the output SQLDA information about a CALL statement, one of the privileges or authorities listed below is required:

- EXECUTE privilege on the stored procedure
- *sysadm* or *dbadm* authority

Required connection:

Database.

Command syntax:



Command parameters:

OUTPUT

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

select-statement, call-statement, or 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.

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. The *schema* is the user name under which the table or view was created.

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. 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.

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

- Index schema
- Index name
- Unique rule
- Column count

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.

SHOW DETAIL

For the DESCRIBE TABLE command, specifies that output include the following additional information as well as a second result set which contains the table data partition expressions (which might return 0 rows if the table is not data partitioned):

- Whether a CHARACTER, VARCHAR or LONG VARCHAR column was defined as FOR BIT DATA
- Column number
- Distribution key sequence
- Code page
- Default
- Table partitioning type (for tables partitioned by range this output appears below the original output)

DESCRIBE using ADMIN_CMD

- Partitioning key columns (for tables partitioned by range this output appears below the original output)

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

- Column names

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

Examples:

Describing the output of a SELECT statement

The following example shows how to describe a SELECT statement:

```
CALL SYSPROC.ADMIN_CMD('describe select * from emp_photo')
```

The following is an example of output for this SELECT statement.

Result set 1

| SQLTYPE_ID | SQLTYPE | SQLLENGTH | SQLSCALE | SQLNAME_DATA | ... |
|------------|-----------|-----------|----------|--------------|-----|
| 452 | CHARACTER | 6 | 0 | EMPNO | ... |
| 448 | VARCHAR | 10 | 0 | PHOTO_FORMAT | ... |
| 405 | BLOB | 102400 | 0 | PICTURE | ... |

3 record(s) selected.

Return Status = 0

Output for this SELECT statement (continued).

| ... | SQLNAME_LENGTH | SQLDATATYPE_NAME_DATA | SQLDATATYPE_NAME_LENGTH | ... |
|-----|----------------|-----------------------|-------------------------|-----|
| ... | 5 | SYSIBM .CHARACTER | 18 | ... |
| ... | 12 | SYSIBM .VARCHAR | 16 | ... |
| ... | 7 | SYSIBM .BLOB | 13 | ... |

...
...
...
...

Describing a table

Describing a non-partitioned table.

```
CALL SYSPROC.ADMIN_CMD('describe table org show detail')
```

The following is an example of output for this CALL statement.

Result set 1

| COLNAME | TYPESHEMA | TYPENAME | FOR_BINARY_DATA | ... |
|----------|-----------|----------|-----------------|-----|
| DEPTNUMB | SYSIBM | SMALLINT | N | ... |
| DEPTNAME | SYSIBM | VARCHAR | N | ... |
| MANAGER | SYSIBM | SMALLINT | N | ... |

DESCRIBE using ADMIN_CMD

```
DIVISION    SYSIBM      VARCHAR    N          ...
LOCATION     SYSIBM      VARCHAR    N          ...
```

5 record(s) selected.

Output for this CALL statement (continued).

```
... LENGTH SCALE NULLABLE COLNO PARTKEYSEQ CODEPAGE DEFAULT
... -----
...      2      0 N          0          1          0 -
...     14      0 Y          1          0     1208 -
...      2      0 Y          2          0          0 -
...     10      0 Y          3          0     1208 -
...     13      0 Y          4          0     1208 -
```

Output for this CALL statement (continued).

Result set 2

```
-----
DATA_PARTITION_KEY_SEQ DATA_PARTITION_EXPRESSION
-----
```

0 record(s) selected.

Return Status = 0

Describing a partitioned table.

```
CALL SYSPROC.ADMIN_CMD('describe table part_table1 show detail')
```

The following is an example of output for this CALL statement.

Result set 1

```
-----
COLNAME      TYPESCHEMA      TYPENAME FOR_BINARY_DATA ...
-----
COL1         SYSIBM          INTEGER N          ...
```

1 record(s) selected.

Output for this CALL statement (continued).

```
... LENGTH SCALE NULLABLE COLNO PARTKEYSEQ CODEPAGE DEFAULT
... -----
...      4      0 N          0          1          0 -
```

Output for this CALL statement (continued).

Result set 2

```
-----
DATA_PARTITION_KEY_SEQ DATA_PARTITION_EXPRESSION
-----
```

1 COL1

1 record(s) selected

Describing a table index

The following example shows how to describe a table index.

```
CALL SYSPROC.ADMIN_CMD('describe indexes for table t1')
```

The following is an example of output for this CALL statement.

Result set 1

```
-----
INDSCHEMA    INDNAME          UNIQUE_RULE      NUMBER_OF_COLUMNS COLNAMES
```


DESCRIBE using ADMIN_CMD

```
-----  
SYSIBM      SQL050117181625680 PRIMARY_INDEX      1 +PK  
TXU         T1_INDEX1          DUPLICATES_ALLOWED 1 +C1
```

2 record(s) selected.

Return Status = 0

Describing a data partition

The following example shows how to describe data partitions.

```
CALL SYSPROC.ADMIN_CMD('describe data partitions for table part_table2')
```

The following is an example of output for this CALL statement.

Result set 1

```
-----  
DATA_PARTITION_ID LOW_KEY_INCLUSIVE LOW_KEY_VALUE ...  
-----  
0 Y                1                ...  
1 Y                10               ...  
2 Y                20               ...
```

3 record(s) selected.

Output for this CALL statement (continued).

```
... HIGH_KEY_INCLUSIVE HIGH_KEY_VALUE  
... -----  
... N                10  
... N                20  
... N                40
```

The following example shows how to describe data partitions with 'SHOW
DETAIL' clause.

```
CALL SYSPROC.ADMIN_CMD('describe data partitions for table part_table2 show detail')
```

The following is an example of output for this CALL statement.

Result set 1

```
-----  
DATA_PARTITION_ID LOW_KEY_INCLUSIVE LOW_KEY_VALUE ...  
-----  
0 Y                1                ...  
1 Y                10               ...  
2 Y                20               ...
```

3 record(s) selected.

Return Status = 0

Output for this CALL statement (continued).

```
... HIGH_KEY_INCLUSIVE HIGH_KEY_VALUE  
... -----  
... N                10  
... N                20  
... N                40
```

Output for this CALL statement (continued).

Result set 2

```
-----
```

```

DATA_PARTITION_ID DATA_PARTITION_NAME TBSPID ...
-----
          0 PART0                3 ...
          1 PART1                3 ...
          2 PART2                3 ...
    
```

3 record(s) selected.

Return Status = 0

Output for this CALL statement (continued).

```

... PARTITION_OBJECT_ID LONG_TBSPID ACCESSMODE STATUS
... -----
...          15                3 FULL_ACCESS
...          16                3 FULL_ACCESS
...          17                3 FULL_ACCESS
    
```

Usage note:

If the DESCRIBE command tries to create a temporary table and fails, creation of SYSTOOLSTMPSPACE is attempted, and then creation of the temporary table is attempted again, this time in SYSTOOLSTMPSPACE. SYSCTRL or SYSADM authority is required to create the SYSTOOLSTMPSPACE table space.

Result set information:

Command execution status is returned in the SQLCA resulting from the CALL statement. If execution is successful, the commands return additional information in result sets as follows:

- Table 24: **DESCRIBE select-statement, DESCRIBE call-statement and DESCRIBE XQUERY XQuery-statement** commands
- Table 25 on page 64: Result set 1 for the **DESCRIBE TABLE** command
- Table 26 on page 65: Result set 2 for the **DESCRIBE TABLE** command
- Table 27 on page 65: **DESCRIBE INDEXES FOR TABLE** command
- Table 28 on page 66: Result set 1 for the **DESCRIBE DATA PARTITIONS FOR TABLE** command
- Table 29 on page 66: Result set 2 for the **DESCRIBE DATA PARTITIONS FOR TABLE** command

Table 24. Result set returned by the DESCRIBE select-statement, DESCRIBE call-statement and DESCRIBE XQUERY XQuery-statement commands

| Column name | Data type | LOB only ¹ | Description |
|-------------|---------------|-----------------------|---|
| SQLTYPE_ID | SMALLINT | No | Data type of the column, as it appears in the SQLTYPE field of the SQL descriptor area (SQLDA). |
| SQLTYPE | VARCHAR (257) | No | Data type corresponding to the SQLTYPE_ID value. |
| SQLLEN | INTEGER | No | Length attribute of the column, as it appears in the SQLLEN field of the SQLDA. |
| SQLSCALE | SMALLINT | No | Number of digits in the fractional part of a decimal value; 0 in the case of other data types. |

DESCRIBE using ADMIN_CMD

Table 24. Result set returned by the DESCRIBE select-statement, DESCRIBE call-statement and DESCRIBE XQUERY XQuery-statement commands (continued)

| Column name | Data type | LOB only ¹ | Description |
|-------------------|---------------|-----------------------|----------------------------|
| SQLNAME_DATA | VARCHAR (128) | No | Name of the column. |
| SQLNAME_LENGTH | SMALLINT | No | Length of the column name. |
| SQLDATA_TYPESHEMA | VARCHAR (128) | Yes | Data type schema name. |
| SQLDATA_TYPENAME | VARCHAR (128) | Yes | Data type name. |

Note: ¹: Yes indicates that non-null values are returned only when there is LOB data being described.

Table 25. Result set 1 returned by the DESCRIBE TABLE command

| Column name | Data type | Detail ² | Description |
|-----------------|---------------|---------------------|---|
| COLNAME | VARCHAR (128) | No | Column name. |
| YPESHEMA | VARCHAR (128) | No | If the column name is distinct, the schema name is returned, otherwise, 'SYSIBM' is returned. |
| TYPENAME | VARCHAR (128) | No | Name of the column type. |
| FOR_BINARY_DATA | CHAR (1) | Yes | Returns 'Y' if the column is of type CHAR, VARCHAR or LONG VARCHAR, and is defined as FOR BIT DATA, 'N' otherwise. |
| LENGTH | INTEGER | No | Maximum length of the data. For DECIMAL data, this indicates the precision. For distinct types, 0 is returned. |
| SCALE | SMALLINT | No | For DECIMAL data, this indicates the scale. For all other types, 0 is returned. |
| NULLABLE | CHAR (1) | No | One of: <ul style="list-style-type: none"> • 'Y' if column is nullable • 'N' if column is not nullable |
| COLNO | SMALLINT | Yes | Ordinal of the column. |
| PARTKEYSEQ | SMALLINT | Yes | Ordinal of the column within the table's partitioning key. NULL or 0 is returned if the column is not part of the partitioning key, and is NULL for subtables and hierarchy tables. |

Table 25. Result set 1 returned by the DESCRIBE TABLE command (continued)

| Column name | Data type | Detail ² | Description |
|-------------|---------------|---------------------|--|
| CODEPAGE | SMALLINT | Yes | Code page of the column and is one of: <ul style="list-style-type: none"> • Value of the database code page for columns that are not defined with FOR BIT DATA. • Value of the DBCS code page for graphic columns. • 0 otherwise. |
| DEFAULT | VARCHAR (254) | Yes | 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. Might also be NULL. |

Note: ²: Yes indicates that non-null values are returned only when the SHOW DETAIL clause is used.

Table 26. Result set 2 returned by the DESCRIBE TABLE command when the SHOW DETAIL clause is used.

| Column name | Data type | Description |
|---------------------------|------------|---|
| DATA_PARTITION_KEY_SEQ | INTEGER | Data partition key number, for example, 1 for the first data partition expression and 2 for the second data partition expression. |
| DATA_PARTITION_EXPRESSION | CLOB (32K) | Expression for this data partition key in SQL syntax |

Table 27. Result set returned by the DESCRIBE INDEXES FOR TABLE command

| Column name | Data type | Detail ² | Description |
|-------------|----------------|---------------------|--|
| INDSCHEMA | VARCHAR (128) | No | Index schema name. |
| INDNAME | VARCHAR (128) | No | Index name. |
| UNIQUE_RULE | VARCHAR (30) | No | One of: <ul style="list-style-type: none"> • DUPLICATES_ALLOWED • PRIMARY_INDEX • UNIQUE_ENTRIES_ONLY |
| COLCOUNT | SMALLINT | No | Number of columns in the key, plus the number of include columns, if any. |
| COLNAMES | VARCHAR (2048) | Yes | List of the column names, each preceded with a + to indicate ascending order or a - to indicate descending order. |

DESCRIBE using ADMIN_CMD

Note: ²: Yes indicates that non-null values are returned only when the SHOW DETAIL clause is used.

Table 28. Result set 1 returned by the DESCRIBE DATA PARTITIONS FOR TABLE command

| Column name | Data type | Detail ² | Description |
|--------------------|---------------|---------------------|---|
| DATA_PARTITION_ID | INTEGER | No | Data partition identifier. |
| LOW_KEY_INCLUSIVE | CHAR (1) | No | 'Y' if the low key value is inclusive, otherwise, 'N'. |
| LOW_KEY_VALUE | VARCHAR (512) | No | Low key value for this data partition. |
| HIGH_KEY_INCLUSIVE | CHAR (1) | No | 'Y' if the high key value is inclusive, otherwise, 'N'. |
| HIGH_KEY_VALUE | VARCHAR (512) | No | High key value for this data partition. |

Note: ²: Yes indicates that non-null values are returned only when the SHOW DETAIL clause is used.

Table 29. Result set 2 returned by the DESCRIBE DATA PARTITIONS FOR TABLE command when the SHOW DETAIL clause is used.

| Column name | Data type | Description |
|---------------------|---------------|---|
| DATA_PARTITION_ID | INTEGER | Data partition identifier. |
| DATA_PARTITION_NAME | VARCHAR (128) | Data partition name. |
| TBSPID | INTEGER | Identifier of the table space where this data partition is stored. |
| PARTITION_OBJECT_ID | INTEGER | Identifier of the DMS object where this data partition is stored. |
| LONG_TBSPID | INTEGER | Identifier of the table space where long data is stored. |
| ACCESSMODE | VARCHAR (20) | Defines accessibility of the data partition and is one of: <ul style="list-style-type: none"> • FULL_ACCESS • NO_ACCESS • NO_DATA_MOVEMENT • READ_ONLY |
| STATUS | VARCHAR(64) | Data partition status and can be one of: <ul style="list-style-type: none"> • NEWLY_ATTACHED • NEWLY_DETACHED: MQT maintenance is required. • INDEX_CLEANUP_PENDING: detached data partition whose tuple in SYSDATAPARTITIONS is maintained only for index cleanup. This tuple is removed when all index records referring to the detached data partition have been deleted. <p>The column is blank otherwise.</p> |

Related concepts:

- “SYSTOOLSPACE and SYSTOOLSTMPSPACE table spaces” in *Administration Guide: Planning*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38

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:

Database. The DAS must be running.

Command syntax:

```
►►—DROP CONTACT—name—————▶▶
```

Command parameters:

CONTACT name

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

Example:

Drop the contact named 'testuser' from the list of contacts on the server system.
CALL SYSPROC.ADMIN_CMD('drop contact testuser')

Usage notes:

The DAS must have been created and be running.

Command execution status is returned in the SQLCA resulting from the CALL statement.

Related reference:

- "ADMIN_CMD – Run administrative commands" on page 38
- "ADD CONTACT command using the ADMIN_CMD procedure" on page 44
- "ADD CONTACTGROUP command using the ADMIN_CMD procedure" on page 46
- "DROP CONTACTGROUP command using the ADMIN_CMD procedure" on page 69
- "UPDATE CONTACT command using the ADMIN_CMD procedure" on page 164
- "UPDATE CONTACTGROUP command using the ADMIN_CMD procedure" on page 166
- "db2admin - DB2 administration server command" in *Command Reference*
- "db2DropContact API - Remove a contact from the list of contacts to whom notification messages can be sent" in *Administrative API Reference*

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:

Database. The DAS must be running.

Command Syntax:

```
►►—DROP CONTACTGROUP—name—————►►
```

Command Parameters:

CONTACTGROUP *name*

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

Example:

```
Drop the contact group named 'gname1'.
CALL SYSPROC.ADMIN_CMD( 'drop contactgroup gname1' )
```

Usage notes:

The DAS must have been created and be running.

Command execution status is returned in the SQLCA resulting from the CALL statement.

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “ADD CONTACT command using the ADMIN_CMD procedure” on page 44
- “ADD CONTACTGROUP command using the ADMIN_CMD procedure” on page 46
- “DROP CONTACT command using the ADMIN_CMD procedure” on page 68
- “UPDATE CONTACT command using the ADMIN_CMD procedure” on page 164
- “UPDATE CONTACTGROUP command using the ADMIN_CMD procedure” on page 166
- “db2admin - DB2 administration server command” in *Command Reference*
- “db2DropContactGroup API - Remove a contact group from the list of contacts to whom notification messages can be sent” in *Administrative API Reference*

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. The data is exported to the server only.

Authorization:

One of the following:

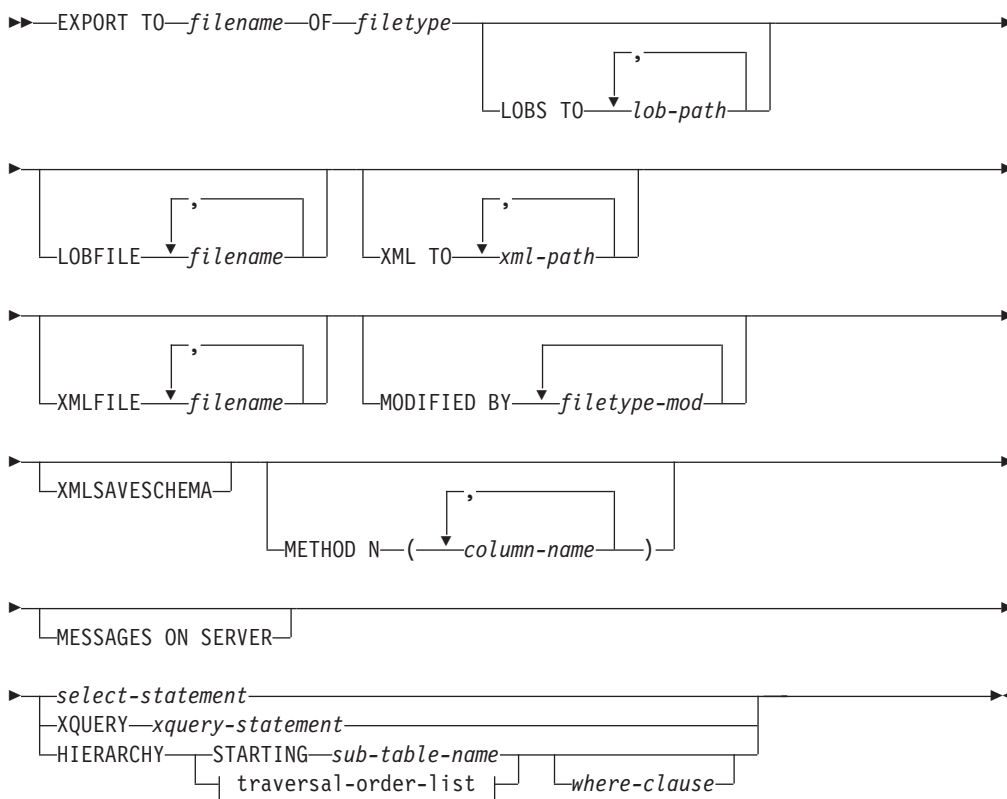
- *sysadm*
- *dbadm*

or CONTROL or SELECT privilege on each participating table or view.

Required connection:

Database. 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:



traversal-order-list:



Command parameters:**HIERARCHY traversal-order-list**

Export a sub-hierarchy 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.

HIERARCHY STARTING sub-table-name

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

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. The maximum number of file names that can be specified is 999. This will implicitly activate the LOBSINFILE behavior.

When creating 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*), and then appending a 3-digit sequence number 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 created will be */u/foo/lob/path/bar.001.lob*, */u/foo/lob/path/bar.002.lob*, and so on.

LOBS TO lob-path

Specifies one or more paths to directories in which the LOB files are to be stored. The path(s) must exist on the coordinator partition of the server and must be fully qualified. There will be at least one file per LOB path, and each file will contain at least one LOB. The maximum number of paths that can be specified is 999. This will implicitly activate the LOBSINFILE behavior.

MESSAGES ON SERVER

Specifies that the message file created on the server by the EXPORT command is to be saved. The result set returned will include the following two columns: *MSG_RETRIEVAL*, which is the SQL statement required to retrieve all the warnings and error messages that occur during this operation, and *MSG_REMOVAL*, which is the SQL statement required to clean up the messages.

If this clause is not specified, the message file will be deleted when the ADMIN_CMD procedure returns to the caller. The *MSG_RETRIEVAL* and *MSG_REMOVAL* column in the result set will contain null values.

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, as well as the directory where the data is to be exported to.

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 WSF and IXF files, but is not valid when exporting hierarchical data.

MODIFIED BY filetype-mod

Specifies file type modifier options. See File type modifiers for the export utility.

EXPORT using ADMIN_CMD

OF filetype

Specifies the format of the data in the output file:

- DEL (delimited ASCII format), which is used by a variety of database manager and file manager programs.
- WSF (work sheet format), which is used by programs such as:
 - Lotus 1-2-3
 - Lotus Symphony

When exporting BIGINT or DECIMAL data, only values that fall within the range of type DOUBLE can be exported accurately. Although values that do not fall within this range are also exported, importing or loading these values back might result in incorrect data, depending on the operating system.

- IXF (integrated exchange format, PC version), in which most of the table attributes, as well as any existing indexes, are saved in the IXF file, except when columns are specified in the SELECT statement. With this format, the table can be recreated, while with the other file formats, the table must already exist before data can be imported into it.

select-statement

Specifies the SELECT or XQUERY statement that will return 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.

TO filename

Specifies the name of the file to which data is to be exported to on the server. This must be a fully qualified path and must exist on the server coordinator partition.

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

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 creating 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), appending a 3-digit sequence number, and appending 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 created will be /u/foo/xml/path/bar.001.xml, /u/foo/xml/path/bar.002.xml, and so on.

XML TO xml-path

Specifies one or more paths to directories in which the XML files are to be stored. There will be at least one file per XML path, and each file will contain at least one XQuery Data Model (QDM) instance. If more than one path is specified, then QDM instances are distributed evenly among the paths.

XMLSAVESHEMA

Specifies that XML schema information should 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 will be 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 XMLSAVESCHEMA option is not compatible with XQuery sequences that do not produce well-formed XML documents.

Example:

The following example shows how to export information from the STAFF table in the SAMPLE database to the file myfile.ixf. The output will be in IXF format. You must be connected to the SAMPLE database before issuing the command.

```
CALL SYSPROC.ADMIN_CMD ('EXPORT to /home/user1/data/myfile.ixf
OF ixf MESSAGES ON SERVER select * from staff')
```

Usage notes:

- Any path used in the EXPORT command must be a valid fully-qualified path on the server.
- If a table contains LOB columns, at least one fully-qualified LOB path and LOB name must be specified, using the LOBS TO and LOBFILE clauses.
- The export utility issues a COMMIT statement at the beginning of the operation which, in the case of Type 2 connections, causes the procedure to return SQL30090N with reason code 2.
- When exporting from a UCS-2 database to a delimited ASCII (DEL) file, all character data is converted to the code page that is in effect where the procedure is executing. Both character string and graphic string data are converted to the same SBCS or MBCS code page of the server.
- Be sure to complete all table operations and release all locks before starting an export operation. This can be done by issuing a COMMIT after closing all cursors opened WITH HOLD, or by issuing a ROLLBACK.
- Table aliases can be used in the SELECT statement.
- The messages placed in the message file include the information returned from the message retrieval service. Each message begins on a new line.
- The export utility produces a warning message whenever a character column with a length greater than 254 is selected for export to DEL format files.
- PC/IXF import should be used to move 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.
- DB2 Connect can be used to export tables from DRDA[®] servers such as DB2 for OS/390[®], DB2 for VM and VSE, and DB2 for OS/400[®]. Only PC/IXF export is supported.
- The export utility will not create multiple-part PC/IXF files when invoked from an AIX system.
- The export utility will store 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.

EXPORT using ADMIN_CMD

- When exporting typed tables, subselect statements can only be expressed by specifying the target table name and the WHERE clause. Fullselect and *select-statement* cannot be specified when exporting a hierarchy.
- For file formats other than IXF, it is recommended that the traversal order list be specified, because it 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 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 exporting data from a table that has protected rows, the LBAC credentials held by the session authorization id might limit the rows that are exported. Rows that the session authorization ID does not have read access to will not be exported. No error or warning is given.
- If the LBAC credentials 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.
- Export packages are bound using 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 using DATETIME LOC format (locale specific format), you may see inconsistent behaviour between CLP and export if the CLP DATETIME format is different from ISO. For instance, the following SELECT statement may 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 using 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
```

Result set information:

Command execution status is returned in the SQLCA resulting from the CALL statement. If execution is successful, the command returns additional information in result sets as follows:

Table 30. Result set returned by the EXPORT command

| Column name | Data type | Description |
|---------------|-----------|--------------------------------|
| ROWS_EXPORTED | BIGINT | Total number of exported rows. |

Table 30. Result set returned by the EXPORT command (continued)

| Column name | Data type | Description |
|---------------|--------------|--|
| MSG_RETRIEVAL | VARCHAR(512) | SQL statement that is used to retrieve messages created by this utility. For example: <pre>SELECT SQLCODE, MSG FROM TABLE (SYSPROC.ADMIN_GET_MSGS ('3203498_txu')) AS MSG</pre> |
| MSG_REMOVAL | VARCHAR(512) | SQL statement that is used to clean up messages created by this utility. For example: <pre>CALL SYSPROC.ADMIN_REMOVE_MSGS ('3203498_txu')</pre> |

Related concepts:

- “Privileges, authorities and authorization required to use export” in *Data Movement Utilities Guide and Reference*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “ADMIN_GET_MSGS table function – Retrieve messages generated by a data movement utility that is executed through the ADMIN_CMD procedure” on page 41
- “ADMIN_REMOVE_MSGS procedure – Clean up messages generated by a data movement utility that is executed through the ADMIN_CMD procedure” on page 43
- “db2Export API - Export data from a database” in *Administrative API Reference*
- “Miscellaneous variables” in *Performance Guide*
- “db2pd - Monitor and troubleshoot DB2 database command” in *Command Reference*

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/sql/lib/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 node (database partition server) in the partitioned database environment.

Authorization:

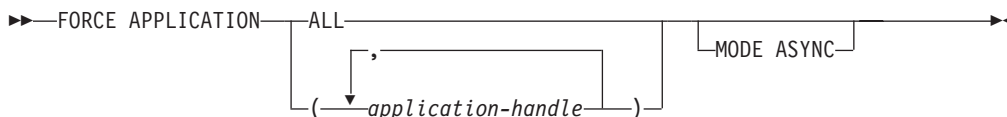
One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required connection:

Database.

Command syntax:



Command parameters:

APPLICATION

ALL All applications will be disconnected from the database. This might close the connection the ADMIN_CMD procedure is running on, which causes an SQL1224N error to be returned for the ADMIN_CMD procedure once the force operation is completed successfully.

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.

FORCE APPLICATION using ADMIN_CMD

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:

```
CALL SYSPROC.ADMIN_CMD( '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.

Users creating a database cannot be forced.

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

Command execution status is returned in the SQLCA resulting from the CALL statement.

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “APPLICATIONS administrative view – Retrieve connected database application information” on page 280
- “sqlfrce API - Force users and applications off the system” in *Administrative API Reference*
- “LIST APPLICATIONS command” in *Command Reference*

GET STMM TUNING DBPARTITIONNUM command using the ADMIN_CMD procedure

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

Authorization:

SYSADM or DBADM authority

Required connection:

Database

Command syntax:

▶▶ GET STMM TUNING DBPARTITIONNUM ◀◀

Example:

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

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 database partition number (USER_PREFERRED_NUMBER) is set by the user and specifies the database partition on which the user wishes to run the memory tuner. While the database is running, the tuning partition is updated asynchronously 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 partition number. 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.

Result set information:

Command execution status is returned in the SQLCA resulting from the CALL statement. If execution is successful, the command returns additional information in the following result set:

GET STMM TUNING DBPARTITIONNUM using ADMIN_CMD

Table 31. Result set returned by the GET STMM TUNING DBPARTITIONNUM command

| Column name | Data type | Description |
|-----------------------|-----------|---|
| USER_PREFERRED_NUMBER | INTEGER | User preferred self tuning memory manager (STMM) tuning database partition number. A value of -1 indicates that the default database partition is used. |
| CURRENT_NUMBER | INTEGER | Current STMM tuning database partition number. A value of -1 indicates that the default database partition is used. |

Related concepts:

- “Using self tuning memory in partitioned database environments” in *Performance Guide*
- “Self tuning memory” in *Performance Guide*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “UPDATE STMM TUNING DBPARTITIONNUM command using the ADMIN_CMD procedure” on page 178

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.

Authorization:

- IMPORT using the INSERT option requires one of the following:
 - *sysadm*
 - *dbadm*
 - 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:
 - *sysadm*
 - *dbadm*
 - 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:
 - *sysadm*
 - *dbadm*
 - 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:
 - *sysadm*
 - *dbadm*
 - 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
 - CREATIN 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:
 - *sysadm*
 - *dbadm*
 - 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:

- *sysadm*
- *dbadm*
- 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 meets these criteria:
 - It is part of the security policy protecting the table
 - It was granted to the session authorization ID for write access

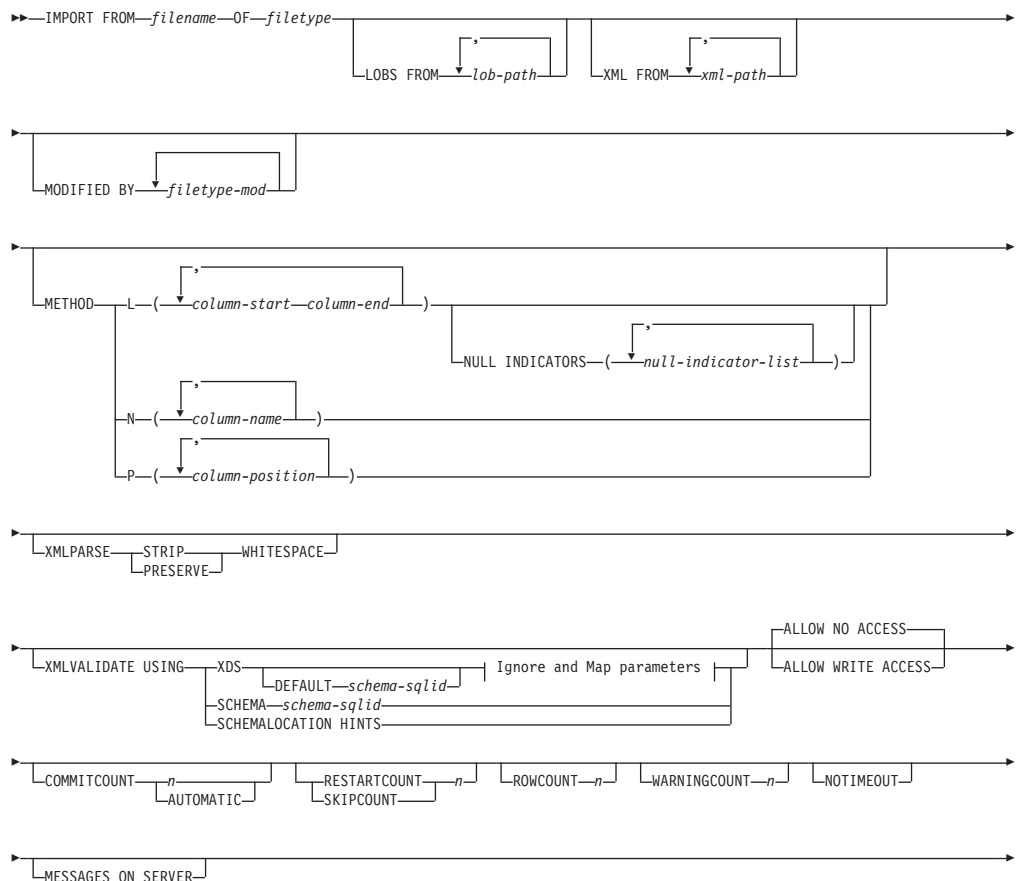
The label on the row to insert, the user's LBAC credentials, the security policy definition, and the LBAC rules determine the label on the row.

- If the REPLACE or REPLACE_CREATE option is specified, the session authorization ID must have the authority to drop the table.

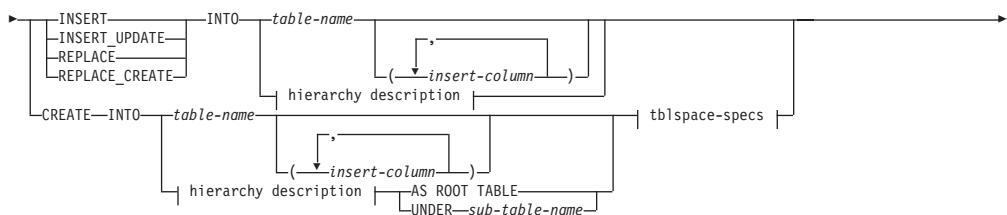
Required connection:

Database. 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.

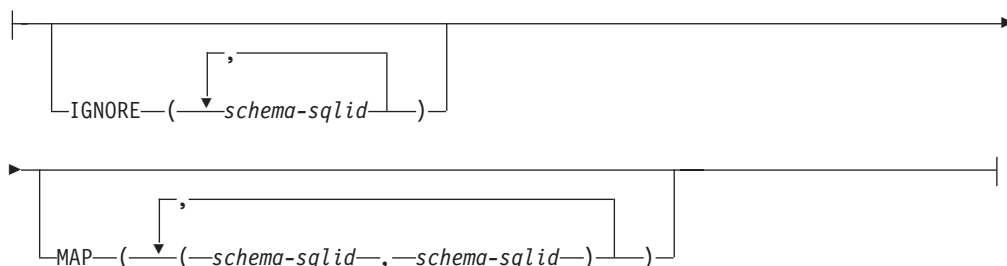
Command syntax:



IMPORT using ADMIN_CMD



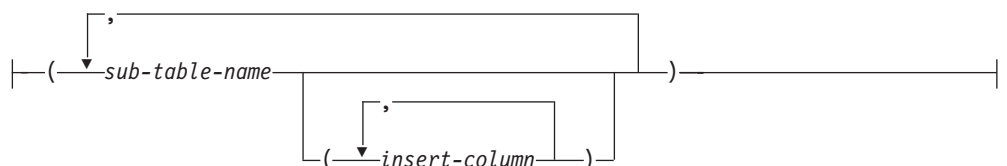
Ignore and Map parameters:



hierarchy description:



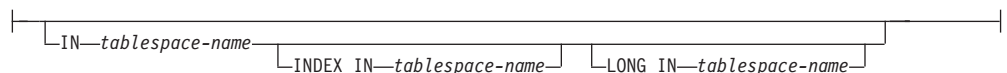
sub-table-list:



traversal-order-list:



tblspace-specs:



Command parameters:

ALL TABLES

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

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).

AS ROOT TABLE

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

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.

If the **IMPORT** command encounters an SQL0964C (Transaction Log Full) while inserting or updating a record and the COMMITCOUNT *n* is specified, **IMPORT** will attempt to resolve the issue by performing an unconditional commit and then reattempt to insert or update the record. If this does not help resolve the log full condition (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. The RESTARTCOUNT or SKIPCOUNT option can be used to avoid processing those row already committed.

CREATE

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 less than 254, CREATE might fail because the rows are too long. See Using import to recreate an exported table for a list of restrictions.

IMPORT using ADMIN_CMD

In this case, the table should be created manually, and IMPORT with INSERT should be invoked, or, alternatively, the LOAD command should be used.

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.

FROM filename

Specifies the name of the file that contains the data to be imported. This must be a fully qualified path and the file must exist on the database server.

HIERARCHY

Specifies that hierarchical data is to be imported.

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.

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. 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.

insert-column

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

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.

INTO table-name

Specifies the database table into which the data is to be imported. This table cannot be a system 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 a down-level 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.

LOBS FROM lob-path

Specifies one or more fully qualified paths that store LOB files. The paths must exist on the database server coordinator partition. 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 behaviour.

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

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.

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.

MESSAGES ON SERVER

Specifies that the message file created on the server by the IMPORT command is to be saved. The result set returned will include the following two columns: MSG_RETRIEVAL, which is the SQL statement required to

IMPORT using ADMIN_CMD

retrieve all the warnings and error messages that occur during this operation, and MSG_REMOVAL, which is the SQL statement required to clean up the messages.

If this clause is not specified, the message file will be deleted when the ADMIN_CMD procedure returns to the caller. The MSG_RETRIEVAL and MSG_REMOVAL column in the result set will contain null values.

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, as well as the directory where the data is to be exported to.

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 to be imported.

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

P Specifies the field numbers of the input data fields to be imported.

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

MODIFIED BY filetype-mod

Specifies file type modifier options. See File type modifiers for the import utility.

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.

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.

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
- WSF (work sheet format), which is used by programs such as:
 - Lotus 1-2-3
 - Lotus Symphony
- IXF (integrated exchange format, PC version), which means it was exported from the same or another DB2 table. An IXF file also contains the table definition and definitions of any existing indexes, except when columns are specified in the SELECT statement.

The WSF file type is not supported when you import to a nickname.

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 honour the CREATE TABLE statement's NOT LOGGED INITIALLY (NLI) clause or the ALTER TABLE statement's ACTIVE NOT LOGGED INITIALLY clause.

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 recreate it, then invoke the import with INSERT statement.

This limitation applies to DB2 UDB Version 7 and DB2 UDB Version 8

REPLACE_CREATE

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 Using import to recreate an exported table 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.

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.

ROWCOUNT *n*

Specifies the number n of physical records in the file to be imported

IMPORT using ADMIN_CMD

(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.

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.

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.

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.

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.

UNDER *sub-table-name*

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

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 desired. 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.

XML FROM *xml-path*

Specifies one or more paths that contain the XML files.

XMLPARSE

Specifies how XML documents are parsed. If this option is not specified, the parsing behaviour for XML documents will be determined by the value of the CURRENT 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 below.

Example:

The following example shows how to import information from the file myfile.ixf to the STAFF table in the SAMPLE database.

```
CALL SYSPROC.AMDIN_CMD
  ('IMPORT FROM /home/userid/data/myfile.ixf
   OF IXF MESSAGES ON SERVER INSERT INTO STAFF')
```

Usage notes:

Any path used in the IMPORT command must be a valid fully-qualified path on the coordinator node for the server.

If the ALLOW WRITE ACCESS or COMMITCOUNT options are specified, a commit will be performed by the import utility. This causes the ADMIN_CMD procedure to return an SQL30090N error with reason code 1 in the case of Type 2 connections.

If the value to be assigned for a column of a result set from the ADMIN_CMD procedure is greater than the maximum value for the data type of the column, then the maximum value for the data type is assigned and a warning message, SQL1155W, is returned.

IMPORT using ADMIN_CMD

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.

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. If automatic COMMITs are not performed, a full log results in a ROLLBACK.

Offline import does not perform automatic COMMITs if any of the following conditions is true:

- the target is a view, not a table
- compound inserts are used
- buffered inserts are used

By default, online import performs automatic COMMITs 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.

One 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 recreate 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 creating 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 60KB), the message file returned to the user on the client might be missing messages from the middle of the import operation. The first 30KB of message information and the last 30KB 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.

The database table or hierarchy must exist before data in the ASC, DEL, or WSF 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

IMPORT using ADMIN_CMD

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 OS/390, 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 declared temporary table, or a summary table.

Views cannot be created through the import utility.

On the Windows operating system:

- Importing logically split PC/IXF files is not supported.
- Importing bad format PC/IXF or WSF 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.

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 wish 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.

Result set information:

Command execution status is returned in the SQLCA resulting from the CALL statement. If execution is successful, the command returns additional information in result sets as follows:

Table 32. Result set returned by the IMPORT command

| Column name | Data type | Description |
|-------------|-----------|---|
| ROWS_READ | BIGINT | Number of records read from the file during import. |

Table 32. Result set returned by the IMPORT command (continued)

| Column name | Data type | Description |
|----------------|--------------|--|
| ROWS_SKIPPED | BIGINT | Number of records skipped before inserting or updating begins. |
| ROWS_INSERTED | BIGINT | Number of rows inserted into the target table. |
| ROWS_UPDATED | BIGINT | Number of rows in the target table updated with information from the imported records (records whose primary key value already exists in the table). |
| ROWS_REJECTED | BIGINT | Number of records that could not be imported. |
| ROWS_COMMITTED | BIGINT | Number of records imported successfully and committed to the database. |
| MSG_RETRIEVAL | VARCHAR(512) | SQL statement that is used to retrieve messages created by this utility. For example: <pre>SELECT SQLCODE, MSG FROM TABLE (SYSPROC.ADMIN_GET_MSGS ('1203498_txu')) AS MSG</pre> |
| MSG_REMOVAL | VARCHAR(512) | SQL statement that is used to clean up messages created by this utility. For example: <pre>CALL SYSPROC.ADMIN_REMOVE_MSGS ('1203498_txu')</pre> |

Related concepts:

- “Privileges, authorities, and authorization required to use import” in *Data Movement Utilities Guide and Reference*

Related tasks:

- “Importing data” in *Data Movement Utilities Guide and Reference*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “ADMIN_GET_MSGS table function – Retrieve messages generated by a data movement utility that is executed through the ADMIN_CMD procedure” on page 41
- “ADMIN_REMOVE_MSGS procedure – Clean up messages generated by a data movement utility that is executed through the ADMIN_CMD procedure” on page 43
- “db2Import API - Import data into a table, hierarchy, nickname or view” in *Administrative API Reference*
- “db2look - DB2 statistics and DDL extraction tool command” in *Command Reference*
- “db2pd - Monitor and troubleshoot DB2 database command” in *Command Reference*

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:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required connection:

Database.

Command syntax:

►►—INITIALIZE TAPE—┬──ON—device──┬──USING—blksize──┬──►►

Command parameters:

ON device

Specifies a valid tape device name. The default value is `\\.\TAPE0`. The device specified must be relative to the server.

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.

Example:

Initialize the tape device to use a block size of 2048 bytes, if the value is within the supported range of block sizes for the device.

```
CALL SYSPROC.ADMIN_CMD( 'initialize tape using 2048' )
```

Usage note:

Command execution status is returned in the SQLCA resulting from the CALL statement.

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “REWIND TAPE command using the ADMIN_CMD procedure” on page 143
- “SET TAPE POSITION command using the ADMIN_CMD procedure” on page 156
- “BACKUP DATABASE command” in *Command Reference*
- “RESTORE DATABASE command” in *Command Reference*
- “BACKUP DATABASE command using the ADMIN_CMD procedure” on page 53

LOAD command using the ADMIN_CMD procedure

Loads data into a DB2 table. Data residing on the server can be in the form of a file, tape, or named pipe. Data can also be loaded from a cursor defined from a query running against the currently connected database or a different database under the same instance, or by using a user-written script or application. If the COMPRESS attribute for the table is set to YES, the data loaded will be subject to compression on every data and database partition for which a dictionary already exists in the table.

Restrictions:

The load utility does not support loading data at the hierarchy level. The load utility is not compatible with range-clustered tables.

Scope:

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

Authorization:

One of the following:

- *sysadm*
- *dbadm*
- load authority on the database and
 - 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 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:
 - It is part of the security policy protecting the table
 - It was granted to the session authorization ID 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. This security label is used to protect a loaded row if the session authorization ID's 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.

- If the REPLACE option is specified, the session authorization ID must have the authority to drop the table.

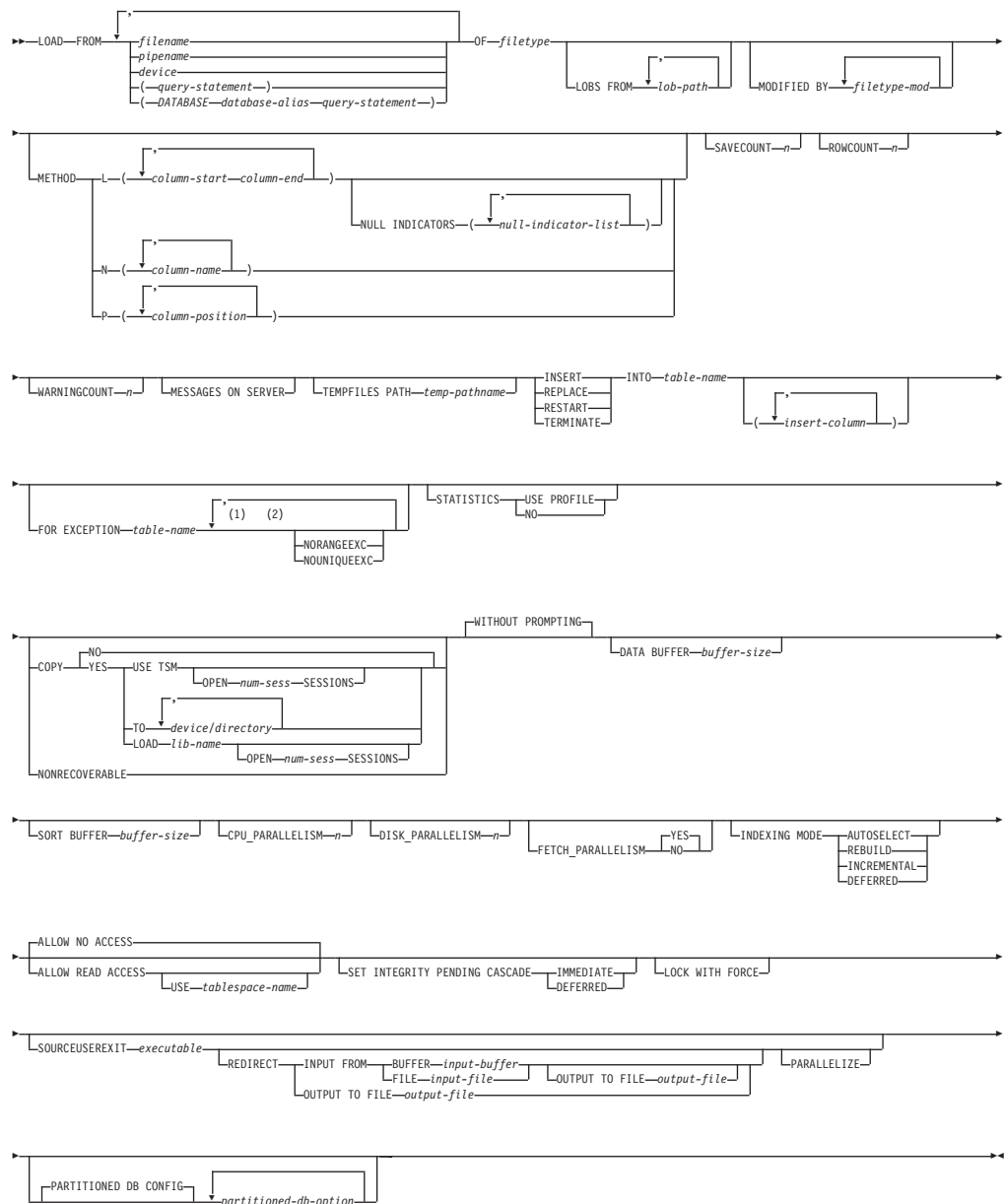
Since all load processes (and all DB2 server processes, in general) are owned by the instance owner, and all of these processes use the identification of the instance owner to access needed files, the instance owner must have read access to input data files. These input data files must be readable by the instance owner, regardless of who invokes the command.

Required connection:

Database.

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:



LOAD using ADMIN_CMD

Notes:

- 1 These keywords can appear in any order.
- 2 Each of these keywords can only appear once.

Command parameters:

FROM filename/pipe/device/(query-statement)/(DATABASE database-alias query-statement)

Specifies the file, pipe or device referring to an SQL statement that contains the data being loaded, or the SQL statement itself and the optional source database to load from cursor.

The *query-statement* option is used to LOAD from a cursor. It contains only one query statement, which is enclosed in parentheses, and can start with VALUES, SELECT or WITH. For example,

```
LOAD FROM (SELECT * FROM T1) OF CURSOR INSERT INTO T2
```

When the *DATABASE database-alias* clause is included prior to the query statement in the parentheses, the LOAD command will attempt to load the data using the *query-statement* from the given database as indicated by the *database-alias* name, which is defined on the server. It must point to a database exist on the server, and is a different database that the application is currently connected to. Note that the LOAD will be executed using the user ID and password explicitly provided for the currently connected database (an implicit connection will cause the LOAD to fail).

If the input source is a file, pipe, or device, it must be accessible from the coordinator partition on the server.

If several names are specified, they will be processed in sequence. If the last item specified is a tape device and the user is prompted for a tape, the LOAD will fail and the ADMIN_CMD procedure will return an error.

Notes:

1. A fully qualified path file name must be used and must exist on the server.
2. 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.
3. 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. (Multiple physical files would be considered logically one if they were all created with one invocation of the EXPORT command.)

OF filetype

Specifies the format of the data:

- ASC (non-delimited ASCII format)
- DEL (delimited ASCII format)
- IXF (integrated exchange format, PC version), exported from the same or from another DB2 table
- CURSOR (a cursor declared against a SELECT or VALUES statement).

LOBS FROM lob-path

The path to the data files containing LOB values to be loaded. The path must end with a slash (/). The path must be fully qualified and accessible from the coordinator partition on the server. 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 behaviour.

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

MODIFIED BY filetype-mod

Specifies file type modifier options. See File type modifiers for the load utility.

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

LOAD using ADMIN_CMD

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.

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 will be monitored using LOAD QUERY. If the value of *n* is not sufficiently high, the synchronization of activities performed at each consistency point will impact performance.

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

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

ROWCOUNT *n*

Specifies the number of *n* physical records in the file to be loaded. Allows a user to load only the first *n* rows in a file.

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 will generate a warning for each row that it attempts to load, which will cause the load to fail. If *n* is zero, or this option is not specified, the load operation will continue regardless of the number of warnings issued. If the load operation is stopped because the threshold of warnings was encountered, another load operation can be started in RESTART mode. The load operation will automatically continue from the last consistency point. Alternatively, another load operation can be initiated in REPLACE mode, starting at the beginning of the input file.

MESSAGES ON SERVER

Specifies that the message file created on the server by the LOAD command is to be saved. The result set returned will include the following two columns: MSG_RETRIEVAL, which is the SQL statement required to retrieve all the warnings and error messages that occur during this operation, and MSG_REMOVAL, which is the SQL statement required to clean up the messages.

If this clause is not specified, the message file will be deleted when the ADMIN_CMD procedure returns to the caller. The MSG_RETRIEVAL and MSG_REMOVAL column in the result set will contain null values.

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.

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. Following is an estimate of how much file system space should be allocated for all temporary files:

- 136 bytes for each message that the load utility generates

- 15KB overhead if the data file contains long field data or LOBs. This quantity can grow significantly if the INSERT option is specified, and there is a large amount of long field or LOB data already in the table.

INSERT

One of four modes under which the load utility can execute. Adds the loaded data to the table without changing the existing table data.

REPLACE

One of four modes under which the load utility can execute. Deletes all existing data from the table, and inserts the loaded data. The table definition and index definitions are not changed. If this option is used when moving data between hierarchies, only the data for an entire hierarchy, not individual subtables, can be replaced.

RESTART

One of four modes under which the load utility can execute. Restarts a previously interrupted load operation. The load operation will automatically continue from the last consistency point in the load, build, or delete phase.

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.

The load terminate option will not remove a backup pending state from table spaces.

INTO table-name

Specifies the database table into which the data is to be loaded. This table cannot be a system table or a declared 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.

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,

```
CALL SYSPROC.ADMIN_CMD('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:

```
CALL SYSPROC.ADMIN_CMD('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 will be copied. Any

LOAD using ADMIN_CMD

row that is in violation of a unique index or a primary key index is copied. If an unqualified table name is specified, the table will be qualified with the CURRENT SCHEMA.

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, on the other hand, contains rows that cannot be loaded because they are invalid or have syntax errors.

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 profile must be created before load is executed. The profile is created by the RUNSTATS command. If the profile does not exist and load is instructed to collect statistics according to the profile, a warning is returned and no statistics are collected.

STATISTICS NO

Specifies that no statistics are to be collected, and that the statistics in the catalogs are not to be altered. This is the default.

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, *logretain* or *userexit* is on). 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 aborts. 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.

COPY YES

Specifies that a copy of the loaded data will be saved. This option is invalid if forward recovery is disabled (both *logretain* and *userexit* are off).

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/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 non-recoverable 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 non-recoverable 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.

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.

This is the default. Any actions which normally require user intervention will return an error message.

DATA BUFFER buffer-size

Specifies the number of 4KB 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.

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 spawn for parsing, converting, and formatting records when building table

LOAD using ADMIN_CMD

objects. This parameter is designed to exploit intra-partition parallelism. 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.

Notes:

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 spawn 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.

FETCH_PARALLELISM YES/NO

When performing 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. 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.

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.

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.

Incremental indexing is not supported when all of the following conditions are true:

- The LOAD COPY option is specified (*logarchmeth1* with the USEREXIT or LOGRETAIN option).
- The table resides in a DMS table space.
- The index object resides in a table space that is shared by other table objects belonging to the table being loaded.

To bypass this restriction, it is recommended that indexes be placed in a separate table space.

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. LOAD TERMINATE or LOAD RESTART of an ALLOW READ ACCESS load can use this option; LOAD TERMINATE or LOAD RESTART of an ALLOW NO ACCESS load cannot use this option. 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.

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 descendents (including descendent foreign key tables, descendent immediate materialized query tables and descendent immediate staging tables).

IMMEDIATE

Indicates that Set Integrity Pending state is immediately extended to all descendent foreign key tables, descendent immediate materialized query tables and descendent staging tables. For a LOAD INSERT operation, Set Integrity Pending state is not extended to descendent 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), descendent 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 descendent foreign key tables, descendent immediate materialized query tables and descendent immediate staging tables will remain unchanged.

Descendent 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). Descendent immediate materialized query tables and descendent immediate staging tables will be implicitly placed in Set Integrity Pending state when one of its underlying tables is checked for integrity violations. A warning (SQLSTATE 01586) will be issued to indicate that dependent tables have been placed in Set Integrity Pending state. See the Notes section of the SET INTEGRITY statement in the SQL Reference for when these descendent 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 descendent foreign key tables, descendent immediate materialized query tables and descendent 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 or SYSCTRL).

ALLOW NO ACCESS loads might force applications holding conflicting locks at the start of the load operation. At the start of the load 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 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 an executable filename which will be called to feed data into the utility.

REDIRECT**INPUT FROM**

BUFFER *input-buffer*

The stream of bytes specified in *input-buffer* is

LOAD using ADMIN_CMD

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

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:

```
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

Reserved.

USING directory

Reserved.

Example:

Issue a load with replace option for the employee table data from a file.

```
CALL SYSPROC.ADMIN_CMD('LOAD FROM /home/theresax/tmp/emp_exp.dat
OF DEL METHOD P (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14)
MESSAGES /home/theresax/tmp/emp_load.msg
REPLACE INTO THERESAX.EMPLOYEE (EMPNO, FIRSTNME, MIDINIT, LASTNAME,
```

LOAD using ADMIN_CMD

```
WORKDEPT, PHONENO, HIREDATE, JOB, EDLEVEL, SEX, BIRTHDATE, SALARY,  
BONUS, COMM) COPY NO INDEXING MODE AUTOSELECT ISOLATE_PART_ERRS  
LOAD_ERRS_ONLY MODE PARTITION_AND_LOAD' )
```

The following is an example of output from a single-partition database.

Result set 1

```
-----  
ROWS_READ      ROWS_SKIPPED    ROWS_LOADED    ROWS_REJECTED    ...  
-----  
          32                0                32                0 ...  
          ...                ...                ...                ...  
          ...                ...                ...                ...  
          1 record(s) selected.
```

Return Status = 0

Output from a single-partition database (continued).

```
... ROWS_DELETED      ROWS_COMMITTED    MSG_RETRIEVAL  
... -----  
...          0                32 SELECT SQLCODE, MSG_TEXT FROM  
...          TABLE(SYSPROC.ADMIN_GET_MSGS(  
...          '2203498_thx')) AS MSG  
...  
...
```

Output from a single-partition database (continued).

```
... MSG_REMOVAL  
... -----  
... CALL SYSPROC.ADMIN_REMOVE_MSGS('2203498_thx')  
...  
...
```

Note: The following columns are also returned in this result set, but are set to NULL because they are only populated when loading into a multi-partition database: ROWS_PARTITIONED and NUM_AGENTINFO_ENTRIES.

The following is an example of output from a multi-partition database.

Result set 1

```
-----  
ROWS_READ      ROWS_REJECTED    ROWS_PARTITIONED    NUM_AGENTINFO_ENTRIES ...  
-----  
          32                0                32                5 ...  
          ...                ...                ...                ...  
          ...                ...                ...                ...  
          ...                ...                ...                ...
```

1 record(s) selected.

Output from a multi-partition database (continued).

```
... MSG_RETRIEVAL      MSG_REMOVAL  
... -----  
... SELECT DBPARTITIONNUM, AGENT_TYPE,    CALL SYSPROC.ADMIN_REMOVE_MSGS  
...   SQLCODE, MSG_TEXT FROM TABLE        ('2203498_thx')  
...   (SYSPROC.ADMIN_GET_MSGS  
...   ('2203498_thx')) AS MSG  
...  
...
```

Note: The following columns are also returned in this result set, but are set to NULL because they are only populated when loading into a single-partition database: ROWS_SKIPPED, ROWS_LOADED, ROWS_DELETED and ROWS_COMMITTED.

LOAD using ADMIN_CMD

Output from a multi-partition database (continued).

Result set 2

| DBPARTITIONNUM | SQLCODE | TABSTATE | AGENTTYPE |
|----------------|---------|----------|---------------|
| 10 | 0 | NORMAL | LOAD |
| 20 | 0 | NORMAL | LOAD |
| 30 | 0 | NORMAL | LOAD |
| 20 | 0 | NORMAL | PARTITION |
| 10 | 0 | NORMAL | PRE_PARTITION |

1 record(s) selected.

Return Status = 0

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.
- 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 prior to loading. Data does not need to be sorted prior to 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.
- The LOAD utility issues a COMMIT statement at the beginning of the operation which, in the case of Type 2 connections, causes the procedure to return SQL30090N with reason code 1.
- Any path used in the LOAD command must be a valid fully-qualified path on the server coordinator partition.
- For performing a load using the CURSOR filetype where the DATABASE keyword was specified during the DECLARE CURSOR command, 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 command). 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** command.

Result set information:

Command execution status is returned in the SQLCA resulting from the CALL statement. If execution is successful, the command returns additional information. A single-partition database will return one result set; a multi-partition database will return two result sets.

- Table 33: Result set for a load operation.
- Table 34 on page 112: Result set 2 contains information for each database partition in a multi-partition load operation.

Table 33. Result set returned by the LOAD command

| Column name | Data type | Description |
|-----------------------|-----------|---|
| ROWS_READ | BIGINT | Number of rows read during the load operation. |
| ROWS_SKIPPED | BIGINT | Number of rows skipped before the load operation started. This information is returned for a single-partition database only. |
| ROWS_LOADED | BIGINT | Number of rows loaded into the target table. This information is returned for a single-partition database only. |
| ROWS_REJECTED | BIGINT | Number of rows that could not be loaded into the target table. |
| ROWS_DELETED | BIGINT | Number of duplicate rows that were not loaded into the target table. This information is returned for a single-partition database only. |
| ROWS_COMMITTED | BIGINT | Total number of rows processed: the number of rows successfully loaded into the target table, plus the number of skipped and rejected rows. This information is returned for a single-partition database only. |
| ROWS_PARTITIONED | BIGINT | Number of rows distributed by all database distributing agents. This information is returned for a multi-partition database only. |
| NUM_AGENTINFO_ENTRIES | BIGINT | Number of entries returned in the second result set for a multi-partition database. This is the number of agent information entries produced by the load operation. This information is returned for multi-partition database only. |

LOAD using ADMIN_CMD

Table 33. Result set returned by the LOAD command (continued)

| Column name | Data type | Description |
|---------------|--------------|--|
| MSG_RETRIEVAL | VARCHAR(512) | SQL statement that is used to retrieve messages created by this utility. For example, <pre>SELECT SQLCODE, MSG FROM TABLE (SYSPROC.ADMIN_GET_MSGS ('2203498_thx')) AS MSG</pre> <p>This information is returned only if the MESSAGES ON SERVER clause is specified.</p> |
| MSG_REMOVAL | VARCHAR(512) | SQL statement that is used to clean up messages created by this utility. For example: <pre>CALL SYSPROC.ADMIN_REMOVE_MSGS ('2203498_thx')</pre> <p>This information is returned only if the MESSAGES ON SERVER clause is specified.</p> |

Table 34. Result set 2 returned by the LOAD command for each database partition in a multi-partition database.

| Column name | Data type | Description |
|----------------|-----------|---|
| DBPARTITIONNUM | SMALLINT | The database partition number on which the agent executed the load operation. |
| SQLCODE | INTEGER | Final SQLCODE resulting from the load processing. |

Table 34. Result set 2 returned by the LOAD command for each database partition in a multi-partition database. (continued)

| Column name | Data type | Description |
|-------------|-------------|---|
| TABSTATE | VARCHAR(20) | <p>Table state after load operation has completed. It is one of:</p> <ul style="list-style-type: none"> • LOADPENDING: Indicates that the load did not complete, but the table on the partition has been left in a LOAD PENDING state. A load restart or terminate operation must be done on the database partition. • NORMAL: Indicates that the load completed successfully on the database partition and the table was taken out of the LOAD IN PROGRESS (or LOAD PENDING) state. Note that the table might still be in Set Integrity Pending state if further constraints processing is required, but this state is not reported by this interface. • UNCHANGED: Indicates that the load did not complete due to an error, but the state of the table has not yet been changed. It is not necessary to perform a load restart or terminate operation on the database partition. <p>Note: Not all possible table states are returned by this interface.</p> |
| AGENTTYPE | VARCHAR(20) | <p>Agent type and is one of:</p> <ul style="list-style-type: none"> • FILE_TRANSFER • LOAD • LOAD_TO_FILE • PARTITIONING • PRE_PARTITIONING |

Related concepts:

- “Privileges, authorities, and authorizations required to use Load” in *Data Movement Utilities Guide and Reference*
- “Load overview” in *Data Movement Utilities Guide and Reference*

Related reference:

- “ADMIN_GET_MSGS table function – Retrieve messages generated by a data movement utility that is executed through the ADMIN_CMD procedure” on page 41
- “ADMIN_REMOVE_MSGS procedure – Clean up messages generated by a data movement utility that is executed through the ADMIN_CMD procedure” on page 43
- “EXPORT command using the ADMIN_CMD procedure” on page 70
- “ADMIN_CMD – Run administrative commands” on page 38

LOAD using ADMIN_CMD

- “Load configuration options for partitioned database environments” in *Data Movement Utilities Guide and Reference*
- “db2pd - Monitor and troubleshoot DB2 database command” in *Command Reference*

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 of the currently connected database partition. Deleting entries from the recovery history file might be necessary if the file becomes excessively large and the retention period is high.

Authorization:

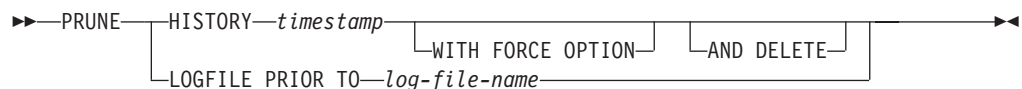
One of the following:

- *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.

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.

LOGFILE PRIOR TO log-file-name

Specifies a string for a log file name, for example *S0000100.LOG*. All log files prior to (but not including) the specified log file will be deleted. The LOGRETAIN database configuration parameter must be set to RECOVERY or CAPTURE.

PRUNE HISTORY/LOGFILE using ADMIN_CMD

Example:

Example 1: Remove all entries from the recovery history file that were written on or before December 31, 2003:

```
CALL SYSPROC.ADMIN_CMD ('prune history 20031231')
```

Example 2: Delete all log files from the active log file path prior to (but not including) S0000100.LOG:

```
CALL SYSPROC.ADMIN_CMD('prune logfile prior to S0000100.LOG')
```

Usage notes:

If the 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 **dbckrst** utility from being able to correctly analyze the complete chain of required backup images. Using the PRUNE HISTORY command without the FORCE option prevents required entries from being deleted.

Pruning backup entries from the history file causes related file backups on DB2 Data Links Manager servers to be deleted.

The command affects only the database partition to which the application is currently connected.

Related concepts:

- “Developing a backup and recovery strategy” in *Data Recovery and High Availability Guide and Reference*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38

QUIESCE DATABASE command using the ADMIN_CMD procedure

Forces all users off the specified database and puts it into a quiesced mode. While the database is in quiesced mode, you can perform administrative tasks on it. After administrative tasks are complete, use the UNQUIESCE command to activate the database and allow other users to connect to the database without having to shut down and perform another database start.

In this mode, only users with authority in this restricted mode are allowed to connect to the database. 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/group and *sysadm*, *sysmaint*, *dbadm*, or *sysctrl* will be able to access the database or its objects.

Authorization:

One of the following:

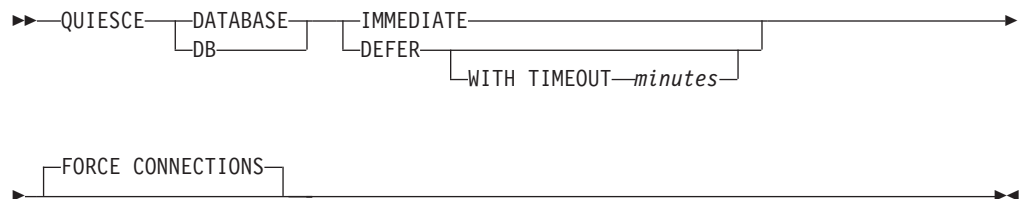
For database level quiesce:

- *sysadm*
- *dbadm*
- *sysadm*
- *sysctrl*

Required connection:

Database

Command syntax:



Command parameters:

DEFER

Wait for applications until they commit the current unit of work.

WITH TIMEOUT

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_timeout* database manager configuration parameter will be used.

QUIESCE DATABASE using ADMIN_CMD

IMMEDIATE

Do not wait for the transactions to be committed, immediately rollback 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.

Example:

Force off all users with connections to the database.

```
CALL SYSPROC.ADMIN_CMD( 'quiesce db immediate' )
```

- This command will force all users off the database if the FORCE CONNECTION option is supplied. FORCE CONNECTION is the default behavior; the parameter is allowed in the command for compatibility reasons.
- The command will be synchronized with the FORCE and will only complete once the FORCE has completed.

Usage notes:

- After QUIESCE DATABASE, users with *sysadm*, *sysmaint*, *sysctrl*, or *dbadm* authority, and GRANT/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>
```

- Command execution status is returned in the SQLCA resulting from the CALL statement.

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “QUIESCE TABLESPACES FOR TABLE command using the ADMIN_CMD procedure” on page 119
- “UNQUIESCE DATABASE command using the ADMIN_CMD procedure” on page 157
- “db2DatabaseQuiesce API - Quiesce the database” in *Administrative API Reference*

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 'N', 'A' or 'D', respectively) are quiesced.

Authorization:

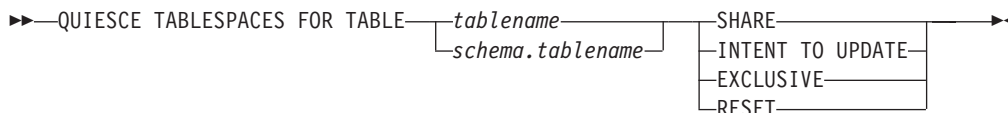
One of the following:

- *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.

QUIESCE TABLESPACES FOR TABLE using ADMIN_CMD

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.

Example:

Quiesce the table spaces containing the staff table.

```
CALL SYSPROC.ADMIN_CMD( 'quiesce tablespaces for table staff share' )
```

Usage notes:

This command is not supported for declared temporary tables.

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 delete pending, quiesce exclusive state. Upon

QUIESCE TABLESPACES FOR TABLE using ADMIN_CMD

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:

```
CALL SYSPROC.ADMIN_CMD('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 upgrade 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.

Command execution status is returned in the SQLCA resulting from the CALL statement.

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “sqluvqdp API - Quiesce table spaces for a table” in *Administrative API Reference*
- “LIST TABLESPACES command” in *Command Reference*
- “QUIESCE DATABASE command using the ADMIN_CMD procedure” on page 117
- “UNQUIESCE DATABASE command using the ADMIN_CMD procedure” on page 157

REDISTRIBUTE DATABASE PARTITION GROUP command using the ADMIN_CMD procedure

Redistributes data across the database partitions in a database partition group. The current data distribution, whether it is uniform or skewed, can be specified. The redistribution algorithm selects the partitions to be moved based on the current data distribution.

Scope:

This command affects all database partitions in the database partition group.

Authorization:

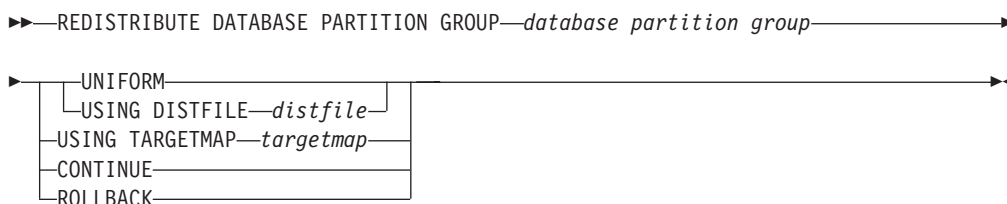
One of the following:

- *sysadm*
- *sysctrl*
- *dbadm*

Required connection:

Connection to the catalog partition.

Command syntax:



Command parameters:

DATABASE PARTITION GROUP *database 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. Tables in the IBMCATGROUP and the IBMTEMPGROUP database partition groups cannot be redistributed.

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 *distfile*

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 *distfile* to indicate the current distribution of data across the 4 096 hash partitions.

REDISTRIBUTE DATABASE PARTITION GROUP using ADMIN_CMD

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 *distfile* 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 database 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 112 000, and partition 3 (with a weight of 0) has no data mapping to it at all.

The *distfile* should contain 4 096 positive integer values in character format. The sum of the values should be less than or equal to 4 294 967 295.

The complete path name for *distfile* must be included and *distfile* must exist on the server and be accessible from the connected partition.

USING TARGETMAP *targetmap*

The file specified in *targetmap* is used as the target distribution map. Data redistribution is done according to this file. The complete path name for *targetmap* must be included and *targetmap* must exist on the server and be accessible from the connected partition.

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 before running REDISTRIBUTE DATABASE PARTITION GROUP.

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 either before or after REDISTRIBUTE DATABASE PARTITION GROUP.

CONTINUE

Continues a previously failed REDISTRIBUTE DATABASE PARTITION GROUP operation. If none occurred, an error is returned.

ROLLBACK

Rolls back a previously failed REDISTRIBUTE DATABASE PARTITION GROUP operation. If none occurred, an error is returned.

Example:

Redistribute database partition group DBPG_1 by providing the current data distribution through a data distribution file, *distfile_for_dbpg_1*, and moving data onto two new database partitions, 6 and 7.

```
CALL SYSPROC.ADMIN_CMD('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 using ADMIN_CMD

Usage notes:

- When a redistribution operation is done, a message file is written to the server to:
 - The /sqllib/redist directory on UNIX based systems, using the following format for subdirectories and file name: *database-name.database-partition-group-name.timestamp*.
 - The \sqllib\redist\ directory on Windows operating systems, using the following format for subdirectories and file name: *database-name\first-eight-characters-of-the-database-partition-group-name\date\time*.
- The time stamp value is the time when the command was issued.
- This utility performs intermittent COMMITs during processing. This can cause type 2 connections to receive an SQL30090N error.
- Use the ALTER DATABASE PARTITION GROUP statement to add database partitions to a database partition group. This statement permits one to define the containers for the table spaces associated with the database partition group.
- DB2 Parallel Edition for AIX Version 1 syntax, with ADD DBPARTITIONNUM and DROP DBPARTITIONNUM options, is supported for users with *sysadm* or *sysctrl* authority. For ADD DBPARTITIONNUM, containers are created like the containers on the lowest node number of the existing nodes within the database partition group.
- 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.
- It is also recommended to update statistics by issuing RUNSTATS after the redistribute database partition group operation has completed.
- Database partition groups containing replicated summary 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 in the database partition group.
- Command execution status is returned in the SQLCA resulting from the CALL statement.
- The file referenced in USING DISTFILE distfile or USING TARGETMAP targetmap, must refer to a file on the server.
- Before starting a redistribute operation, ensure there are no tables in the Load Pending state. Table states can be checked by using the **LOAD QUERY** command. If you discover data in the wrong database partition as a result of a redistribute operation, there are two options. You can:
 1. unload the table, drop it and then reload the table, or
 2. use a new target map to redistribute the database partition group again.

Compatibilities:

For compatibility with versions earlier than Version 8:

- The keyword NODEGROUP can be substituted for DATABASE PARTITION GROUP.

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38

REDISTRIBUTE DATABASE PARTITION GROUP using ADMIN_CMD

- “sqludrdt API - Redistribute data across a database partition group” in *Administrative API Reference*
- “LIST DATABASE DIRECTORY command” in *Command Reference*
- “RUNSTATS command” in *Command Reference*
- “REBIND command” in *Command Reference*

REORG INDEXES/TABLE command using the ADMIN_CMD procedure

Reorganizes an index or a table.

You can reorganize all indexes defined on a table by rebuilding the index data into unfragmented, physically contiguous pages. Alternatively, you have the option of reorganizing specific indexes on a range partitioned table.

If you specify the CLEANUP ONLY option of the index clause, cleanup is performed without rebuilding the indexes. This command cannot be used against indexes on declared temporary tables (SQLSTATE 42995).

The table option reorganizes a table by reconstructing the rows to eliminate fragmented data, and by compacting information.

Scope:

This command affects all database partitions in the database partition group.

Authorization:

One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- CONTROL privilege on the table.

Required connection:

Database

Command syntax:

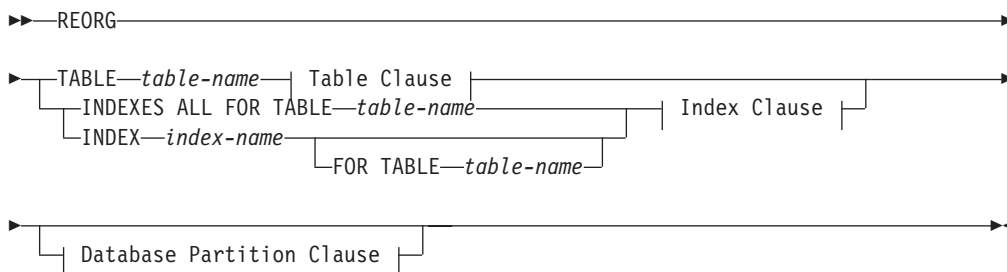
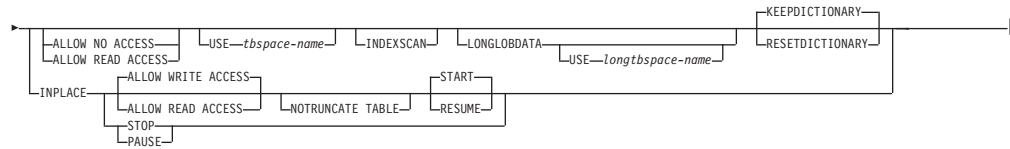


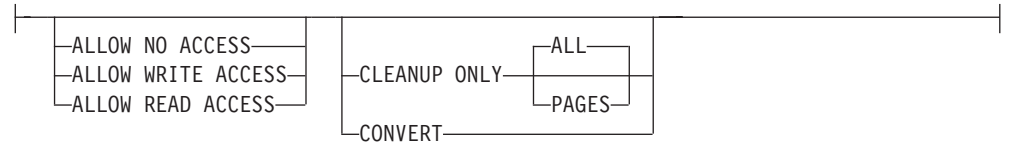
Table Clause:



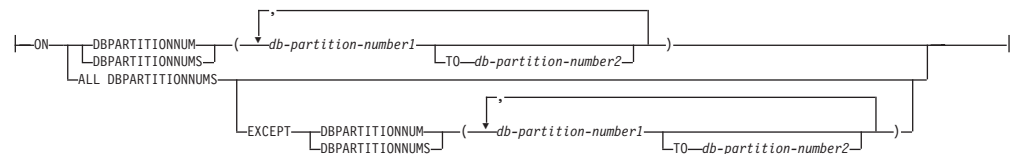
REORG INDEXES/TABLE using ADMIN_CMD



Index Clause:



Database Partition 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 on a partitioned table. Reorganization of individual indexes are ONLY supported for non-partitioned indexes on a partitioned table. This parameter is not supported for block indexes.

FOR TABLE table-name

Specifies the table name location of the individual index being reorganized on a partitioned table. This parameter is optional, given that index names are unique across the database.

ALLOW NO ACCESS

Specifies that no other users can access the table while the indexes are being reorganized.

ALLOW READ ACCESS

Specifies that other users can have read-only access to the table while the indexes are being reorganized. This access level is not supported for REORG INDEXES of a partitioned table unless the CLEANUP ONLY option is specified.

ALLOW WRITE ACCESS

Specifies that other users can read from and write to the table while the indexes are being reorganized. This access level is not supported for multi-dimensionally clustered (MDC) tables, partitioned tables, extended indexes, or tables containing a column with the XML data type unless the CLEANUP ONLY option is specified.

When no ACCESS mode is specified, one will be chosen for you in the following way:

REORG INDEXES/TABLE using ADMIN_CMD

Table 35. Default table access chosen based on the command, table type and additional parameters specified for the index clause:

| Command | Table type | Additional parameters specified for index clause | Default access mode |
|---------------|-----------------------|--|---------------------|
| REORG INDEXES | non-partitioned table | any | ALLOW READ ACCESS |
| REORG INDEXES | partitioned table | none specified | ALLOW NO ACCESS |
| REORG INDEXES | partitioned table | CLEANUP ONLY specified | ALLOW READ ACCESS |
| REORG INDEX | partitioned table | any | ALLOW READ ACCESS |

CLEANUP ONLY

When CLEANUP ONLY is requested, a cleanup rather than a full reorganization will be done. The indexes will not be rebuilt and any pages freed up will be available for reuse by indexes defined on this table only.

The CLEANUP ONLY PAGES option will search 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 indexes can be determined by running runstats and looking at the NUM EMPTY LEAFS column in SYSCAT.INDEXES. The PAGES option will clean the NUM EMPTY LEAFS if they are determined to be committed.

The CLEANUP ONLY ALL option will free committed pseudo empty pages, as well as remove committed pseudo deleted keys from pages that are not pseudo empty. This option will also try to merge adjacent leaf pages if doing so will result in a merged leaf page that has at least PCTFREE free space on the merged leaf page, where PCTFREE is the percent free space defined for the index at index creation time. The default PCTFREE is ten percent. If two pages can be merged, one of the pages will be freed. The number of pseudo deleted keys in an index, excluding those on pseudo empty pages, can be determined by running runstats and then selecting the NUMRIDS DELETED from SYSCAT.INDEXES. The ALL option will clean the NUMRIDS DELETED and the NUM EMPTY LEAFS if they are determined to be committed.

ALL Specifies that indexes should be cleaned up by removing committed pseudo deleted keys and committed pseudo empty pages.

PAGES

Specifies that committed pseudo empty pages should be removed from the index tree. This will not clean up pseudo deleted keys on pages that are not pseudo empty. Since it is only checking the pseudo empty leaf pages, it is considerably faster than using the ALL option in most cases.

CONVERT

If you are not sure whether the table you are operating on has a type-1 or type-2 index, but want type-2 indexes, you can use the CONVERT option. If the index is type 1, this option will convert it into type 2. If the index is already type 2, this option has no effect.

All indexes created by DB2 prior to Version 8 are type-1 indexes. All indexes created by Version 8 are Type 2 indexes, except when

you create an index on a table that already has a type 1 index. In this case the new index will also be of type 1.

Using the INSPECT command to determine the index type can be slow. CONVERT allows you to ensure that the new index will be Type 2 without your needing to determine its original type.

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. While ALLOW READ ACCESS and ALLOW WRITE ACCESS allow access to the table, during the period in which the reorganized copies of the indexes are made available, no access to the table is allowed.

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.

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) table. In place reorganization of tables cannot be used for MDC tables.

INDEX index-name

Specifies the index to use when reorganizing 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 clustering index. If the in place option is not specified, any index specified will be used. If you do not specify the name of an index, the records are reorganized without regard to order. If the table has a clustering index defined, however, 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 table.

ALLOW NO ACCESS

Specifies that no other users can access the table while the table is being reorganized. When reorganizing a partitioned table, this is the default. Reorganization of a partitioned table occurs offline.

ALLOW READ ACCESS

Allow only read access to the table during reorganization. This is the default for a non-partitioned table.

INPLACE

Reorganizes the table while permitting user access.

In place table reorganization is allowed only on non-partitioned and non-MDC tables with type-2 indexes, but without extended indexes and with no indexes defined over XML columns in the table. In place table reorganization takes place asynchronously, and might not be effective immediately.

REORG INDEXES/TABLE using ADMIN_CMD

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.

NOTTRUNCATE 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 is the default, this keyword is optional.

STOP Stop the in place REORG processing at its current point.

PAUSE

Suspend or pause in place REORG for the time being.

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 while resuming.

USE *tblspace-name*

Specifies the name of a system temporary table space in which to store a temporary copy of the table being 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 being reorganized.

For an 8KB, 16KB, or 32KB 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.

When you have two temporary tablespaces of the same page size, and you specify one of them in the USE clause, they will be used in a round robin fashion if there is an index in the table being reorganized. Say you have two tablespaces, *tempspace1* and *tempspace2*, both of the same page size and you specify *tempspace1* in the **REORG** command with the USE option. When you perform **REORG** the first time, *tempspace1* is used. The second time, *tempspace2* is used. The third time, *tempspace1* is used and so on. To avoid this, you should drop one of the temporary tablespaces.

For partitioned tables, the table space is used as temporary storage for the reorganization of all the data partitions in the table. Reorganization of a partitioned table reorganizes a single data partition at a time. The amount of space required is equal to the largest data partition in the table, and not the entire table.

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.

INDEXSCAN

For a clustering REORG an index scan will be used to re-order 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, 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.

LONGLOBDATA

Long field and LOB data are to be reorganized.

This 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.

USE longtbspace-name

This is an optional parameter, which can be used to specify the name of a temporary tablespace to be used for rebuilding long data. If no temporary tablespace is specified for either the table object or for the long objects, the objects will be constructed in the tablespace they currently reside. If a temporary tablespace is specified for the table but this parameter is not specified, then the tablespace used for base reorg data will be used, unless the page sizes differ. In this situation, the DB2 database system will attempt to choose a temporary container of the appropriate page size to create the long objects in.

If USE-longtbspace is specified, USE-tbspace must also be specified. If it is not, the longtbspace 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 processed during reorganization are subject to compression using the existing dictionary. If the COMPRESS attribute for the table is NO and the table has a compression dictionary then reorg processing will remove the dictionary and all the rows in the newly reorganized table will be in non-compressed format. It is not possible to compress long, LOB, index, or XML objects.

Table 36. 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 | |

REORG INDEXES/TABLE using ADMIN_CMD

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 will save the dictionary and not discard it. The dictionary is logged in its entirety for recovery purposes and for future support with data capture changes (i.e. replication).

RESETDICTIONARY

If the COMPRESS attribute for the table is YES then a new row compression dictionary is built. All the rows processed during reorganization are subject to compression 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 will remove the dictionary and all rows in the newly reorganized table will be in non-compressed format. It is not possible to compress long, LOB, index, or XML objects.

Table 37. REORG RESETDICTIONARY

| Compress | Dictionary Exists | Result; outcome | |
|----------|-------------------|--|--|
| Y | Y | Build new dictionary*; rows compressed | |
| Y | N | Build new dictionary; rows compressed | |
| N | Y | Remove dictionary; all rows uncompressed | |
| N | N | No effect; all rows uncompressed | |

* - If a dictionary exists and the compression attribute is enabled but there is currently insufficient data in the table to build a new dictionary, the RESETDICTIONARY operation will keep the existing dictionary. Rows which are smaller in size than the internal minimum record length and rows which do not demonstrate a savings in record length when an attempt is made to compress them are considered 'insufficient' in this case.

Example:

Reorganize the tables in a database partition group consisting of database partitions 1, 3 and 4.

```
CALL SYSPROC.ADMIN_CMD ('REORG TABLE employee  
INDEX empid ON DBPARTITIONNUM (1,3,4)')
```

Usage notes:

Restrictions:

- Command execution status is returned in the SQLCA resulting from the CALL statement.
- The REORG utility issue a COMMIT statement at the beginning of the operation which, in the case of Type 2 connections, causes the procedure to return SQL30090N with reason code 2.

- The **REORG** utility does not support the use of nicknames.
- The **REORG TABLE** command is not supported for declared 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 inplace reorg is not allowed on the table.
- For partitioned tables:
 - REORG is supported at the table level. Reorganization of an individual data partition can be achieved by detaching the data partition, reorganizing the resulting non-partitioned table and then re-attaching the data partition.
 - 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 an error occurs, the non-partitioned indexes of the table are marked bad, and are rebuilt on the next access to the table.
 - If a reorganization operation fails, some data partitions may be in a reorganized state and others may not. When the REORG TABLE command is reissued, all the data partitions will be reorganized regardless of the data partition's reorganization state.
 - When reorganizing indexes on partitioned tables, it is recommended that you perform a runstats operation after asynchronous index cleanup is complete in order to generate accurate index statistics in the presence of detached data partitions. To determine whether or not there are detached data partitions in the table, you can check the status field in SYSDATAPARTITIONS and look for the value "I" (index cleanup) or "D" (detached with dependant MQT).

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 an indexed table has been modified 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 using the index, and can impact the effectiveness of index page prefetching. REORG INDEX or REORG INDEXES can be used to reorganize one or all of the indexes on a table. Index reorganization will remove any fragmentation and restore physical clustering to the leaf pages. Use REORGCHK to help determine if an index needs reorganizing. Be sure to complete all database operations and release all locks before invoking index reorganization. This can be done by issuing a COMMIT after closing all cursors opened WITH HOLD, or by issuing a ROLLBACK.

REORG INDEXES/TABLE using ADMIN_CMD

Indexes might not be optimal following an in-place REORG TABLE operation, since only the data object and not the indexes are reorganized. It is recommended that you perform a REORG INDEXES after an in place REORG TABLE operation. Indexes are completely rebuilt during the last phase of a classic REORG TABLE, however, so reorganizing indexes is not necessary.

Tables that have been modified so many times that data is fragmented and access performance is noticeably slow are candidates for the REORG TABLE command. You should also invoke this utility after altering the inline length of a structured type column in order to benefit from the altered inline length. Use REORGCHK to determine whether a table needs reorganizing. Be sure to complete all database operations and release all locks before invoking REORG TABLE. This can be done by issuing a COMMIT after closing all cursors opened WITH HOLD, or by issuing a ROLLBACK. After reorganizing a table, use RUNSTATS to update the table statistics, and REBIND to rebind the packages that use this table. The reorganize utility will implicitly close all the cursors.

If the table contains mixed row format because the table value compression has been 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 will have the table or index reorganization rolled back.

If the reorganization is not successful, temporary files should 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 will be ordered according to the clustering index.

The PCTFREE value of a table determines the amount of free space designated per page. If the value has not been set, the utility will fill up as much space as possible on each page.

To complete a table space roll-forward recovery following a table reorganization, both regular and large table spaces must be enabled for roll-forward 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 can be a result of the order in which the rows were reorganized, and the types of table spaces used (SMS or DMS).

Related concepts:

- “Table reorganization” in *Performance Guide*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “RUNSTATS command using the ADMIN_CMD procedure” on page 144
- “REORGCHK_TB_STATS procedure – Retrieve table statistics for reorganization evaluation” on page 553
- “REORGCHK_IX_STATS procedure – Retrieve index statistics for reorganization evaluation” on page 550

REORG INDEXES/TABLE using ADMIN_CMD

- “SNAPTAB_REORG administrative view and SNAP_GET_TAB_REORG table function – Retrieve table reorganization snapshot information” on page 436
- “db2Reorg API - Reorganize an index or a table” in *Administrative API Reference*
- “GET SNAPSHOT command” in *Command Reference*
- “REORGCHK command” in *Command Reference*
- “REBIND command” in *Command Reference*

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.

Authorization:

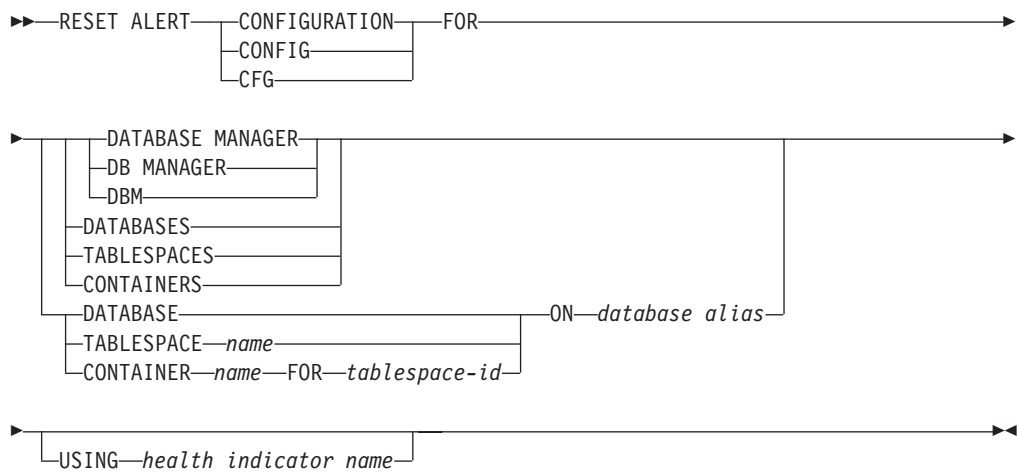
One of the following:

- sysadm
- sysmaint
- sysctrl

Required connection:

Database.

Command syntax:



Command parameters:

DATABASE MANAGER

Resets alert settings for the database manager.

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.

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 name ON database alias"` clause.

CONTAINER name FOR tablespace-id FOR tablespace-id ON database alias

Resets the alert settings for the table space container called `name`, for the table space specified using the `"FOR tablespace-id"` clause, on the database

RESET ALERT CONFIGURATION using ADMIN_CMD

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.

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 *name* ON *database alias*" clause.

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 *name* ON *database alias*

Resets 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 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.

Example:

Reset alert settings for the database manager that owns the database which contains the ADMIN_CMD procedure.

```
CALL SYSPROC.ADMIN_CMD( 'reset alert cfg for dbm' )
```

Usage notes:

Command execution status is returned in the SQLCA resulting from the CALL statement.

The *database alias* must be a local database defined in the catalog on the server because the ADMIN_CMD procedure runs on the server only.

Related tasks:

- "Configuring health indicators using a client application" in *System Monitor Guide and Reference*

Related reference:

- "ADMIN_CMD – Run administrative commands" on page 38
- "UPDATE ALERT CONFIGURATION command using the ADMIN_CMD procedure" on page 159
- "HEALTH_GET_ALERT_ACTION_CFG table function –Retrieve health alert action configuration settings" on page 223
- "HEALTH_GET_ALERT_CFG table function – Retrieve health alert configuration settings" on page 226

RESET ALERT CONFIGURATION using ADMIN_CMD

- “db2ResetAlertCfg API - Reset the alert configuration of health indicators” in *Administrative API Reference*

RESET DATABASE CONFIGURATION command using the ADMIN_CMD procedure

Resets the configuration of a specific database to the system defaults.

Scope:

This command only affects the database partition that the application is connected to.

Authorization:

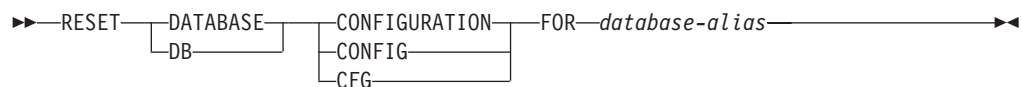
One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required connection:

Database.

Command syntax:



Command parameters:

FOR database-alias

Specifies the alias of the database whose configuration is to be reset to the system defaults. The database alias must be one that is defined in the catalog on the server, and must refer to a local database on the server.

Example:

Reset the configuration of a database cataloged with alias SAMPLE on the server
`CALL SYSPROC.ADMIN_CMD('reset db cfg for SAMPLE')`

Usage notes:

To view or print a list of the database configuration parameters, use the SYSIBMADM.DBCFG administration view.

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.

RESET DATABASE CONFIGURATION using ADMIN_CMD

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 pre-database configuration values, where `AUTO_RUNSTATS` will be ON. `SELF_TUNING_MEMORY` will be reset to ON on non-partitioned database environments and to OFF on partitioned database environments.

Command execution status is returned in the SQLCA resulting from the CALL statement.

The *database alias* must be a local database defined in the catalog on the server because the ADMIN_CMD procedure runs on the server only.

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “RESET DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure” on page 141
- “UPDATE DATABASE CONFIGURATION command using the ADMIN_CMD procedure” on page 168
- “UPDATE DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure” on page 171
- “DBCFCG administrative view – Retrieve database configuration parameter information” on page 182
- “GET DATABASE CONFIGURATION command” in *Command Reference*
- “Configuration parameters summary” in *Performance Guide*
- “db2CfgSet API - Set the database manager or database configuration parameters” in *Administrative API Reference*

RESET DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure

Resets the parameters in the database manager configuration file to the system defaults for the instance that contains the currently connected database. The values are reset by node type.

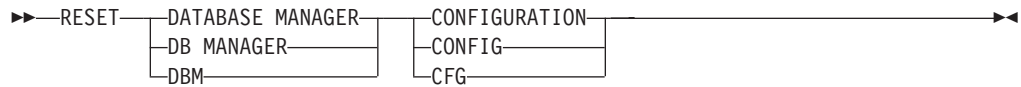
Authorization:

sysadm

Required connection:

Database.

Command syntax:



Command parameters:

None

Example:

Reset the configuration of the instance which contains the database the ADMIN_CMD stored procedure belongs to.

```
CALL SYSPROC.ADMIN_CMD( 'reset dbm cfg' )
```

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 SVCENAME 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 SYSIBMADM.DBMCFG administrative view 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 through the ADMIN_CMD procedure.

It is not recommended that the SVCENAME 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 SYSIBMADM.DBMCFG administration view. To change the value of a configurable parameter, use the UPDATE DATABASE MANAGER CONFIGURATION command through the ADMIN_CMD procedure.

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 on which parameters are configurable on-line and which ones are not, see the configuration parameter summary. Server configuration parameters that are not reset immediately are reset

RESET DATABASE MANAGER CONFIGURATION using ADMIN_CMD

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 the database manager you edit the configuration file manually 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

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “RESET DATABASE CONFIGURATION command using the ADMIN_CMD procedure” on page 139
- “UPDATE DATABASE CONFIGURATION command using the ADMIN_CMD procedure” on page 168
- “UPDATE DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure” on page 171
- “DBMCFG administrative view – Retrieve database manager configuration parameter information” on page 184

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:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required connection:

Database.

Command syntax:

```

▶▶—REWIND TAPE—┐
                  └ON—device—┘
  
```

Command parameters:

ON device

Specifies a valid tape device name. The default value is `\\.\TAPE0`. The device specified must be relative to the server.

Example:

```

Rewind the tape on the device named '\\.\TAPE1'.
CALL SYSPROC.ADMIN_CMD( 'rewind tape on '\\.\TAPE1' )
  
```

Usage note:

Command execution status is returned in the SQLCA resulting from the CALL statement.

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “INITIALIZE TAPE command using the ADMIN_CMD procedure” on page 94
- “SET TAPE POSITION command using the ADMIN_CMD procedure” on page 156

RUNSTATS command using the ADMIN_CMD procedure

Updates statistics about the characteristics of a table and/or associated indexes, or statistical views. These characteristics include number of records, number of pages, and average record length. The optimizer uses these statistics when determining access paths to the data.

For a table, this utility should be called when the table has had many updates, or after reorganizing the table. For a statistical view, this utility should be called when changes to underlying tables have substantially affected the rows returned by the view. The view must have been previously enabled for use in query optimization using the **ALTER VIEW** command.

Scope:

This command can be issued from any database partition in the `db2nodes.cfg` file. It can be used 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 using data from tables on all participating database partitions.

Authorization:

For tables, one of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- CONTROL privilege on the table
- LOAD authority

You do not need any explicit privilege to use this command on any declared global temporary table that exists within its connection.

For statistical views, one of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- CONTROL privilege on the statistical view

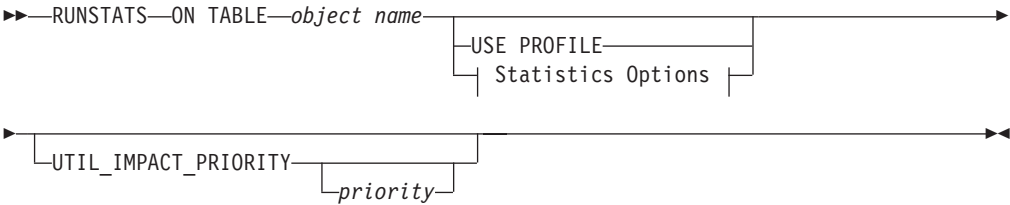
In addition, you need to have appropriate privileges to access rows from the statistical view. Specifically, for each table, statistical view or nickname referenced in the statistical view definition, the user must have one of the following privileges:

- *sysadm* or *dbadm*
- CONTROL
- SELECT

Required connection:

Database

Command syntax:



Statistics Options:

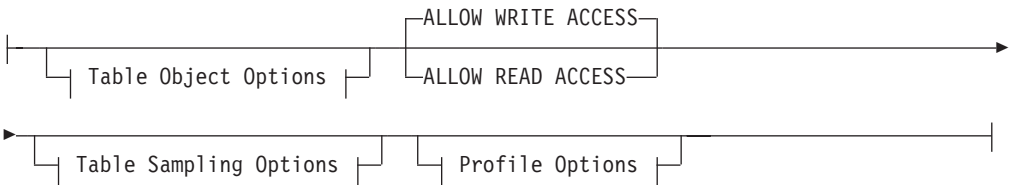


Table Object Options:

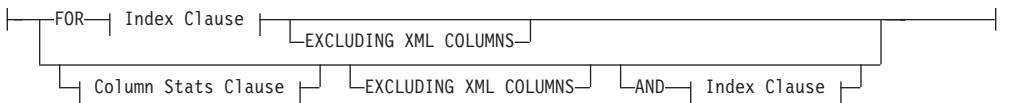
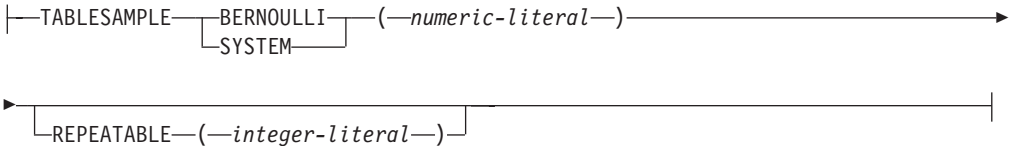


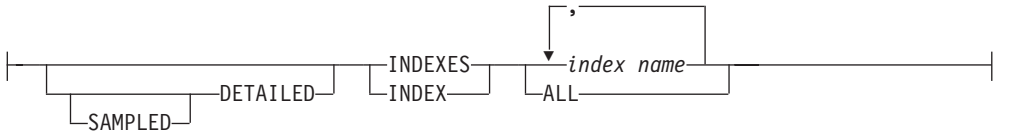
Table Sampling Options:



Profile Options:

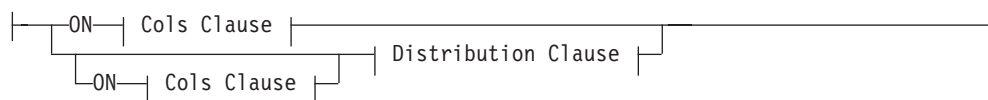


Index Clause:

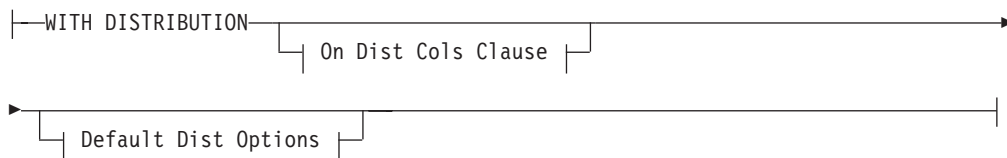


RUNSTATS using ADMIN_CMD

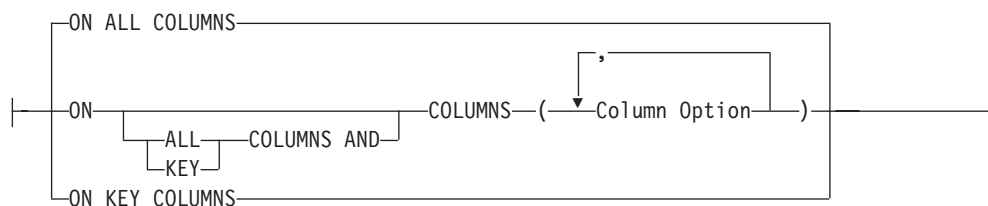
Column Stats Clause:



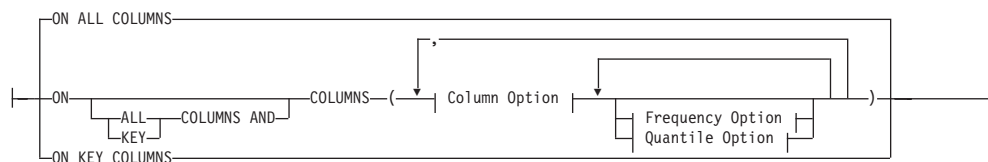
Distribution Clause:



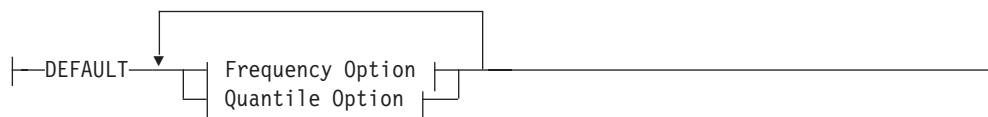
On Cols Clause:



On Dist Cols Clause:



Default Dist Option:



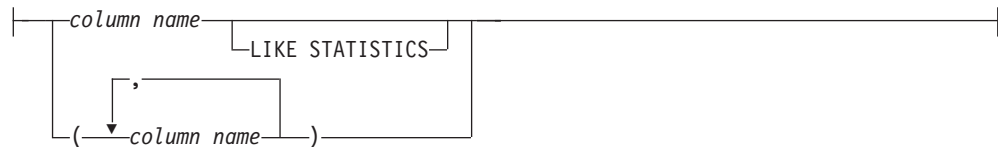
Frequency Option:



Quantile Option:



Column Option:



Command parameters:

object-name

Identifies the table or statistical view on which statistics are to be collected. It must not be a hierarchy table. For typed tables, object-name 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.

index-name

Identifies an existing index defined on the table. The fully qualified name in the form *schema.index-name* must be used. This option cannot be used for views.

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 using the SET PROFILE options and is updated using the UPDATE PROFILE options.

FOR INDEXES

Collects and updates statistics for the indexes only. If no table statistics had been previously collected on the table, basic table statistics are also collected. These basic statistics do not include any distribution statistics. This option cannot be used for views.

AND INDEXES

Collects and updates statistics for both the table and the indexes. This option cannot be used for views.

DETAILED

Calculates extended index statistics. These are the CLUSTERFACTOR and PAGE_FETCH_PAIRS statistics that are gathered for relatively large indexes. This option cannot be used for views.

SAMPLED

This option, when used with the DETAILED option, allows **RUNSTATS** to employ a CPU sampling technique when compiling the extended index statistics. If the option is not specified, every entry in the index is examined to compute the extended index statistics. This option cannot be used for views.

ON ALL COLUMNS

Statistics collection can be done on some columns and not on others. Columns such as LONG VARCHAR or CLOB columns are ineligible. If it is desired to collect statistics on all eligible columns, one can use the ON ALL COLUMNS clause. Columns can be specified either for basic statistics collection (*on-cols-clause*) or in conjunction with the WITH DISTRIBUTION clause (*on-dist-cols-clause*). The ON ALL COLUMNS specification is the default option if neither of the column specific clauses are specified.

If it is specified in the *on-cols-clause*, all columns will have only basic column statistics collected unless specific columns are chosen as part of the

RUNSTATS using ADMIN_CMD

WITH DISTRIBUTION clause. Those columns 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. Anything specified in the on-cols-clause is redundant and therefore not necessary.

ON COLUMNS

This clause allows the user to specify a list of columns for which to collect statistics. If you specify group of columns, the number of distinct values for the group will be collected. When you run **RUNSTATS** on a table without gathering index statistics, and specify a subset of columns for which statistics are to be gathered, then:

1. Statistics for columns not specified in the **RUNSTATS** command but which are the first column in an index are NOT reset.
2. Statistics for all other columns not specified in the **RUNSTATS** command are reset.

This clause can be used in the on-cols-clause and the on-dist-cols-clause. Collecting distribution statistics for a group of columns is not currently supported.

If XML type columns are specified in a column group, the XML type columns will be ignored for the purpose of collecting distinct values for the group. However, basic XML column statistics will be collected for the XML type columns in the column group.

EXCLUDING XML COLUMNS

This clause allows you to omit all XML type columns from statistics collection. 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 will be ignored during statistics collection.

ON KEY COLUMNS

Instead of listing specific columns, you can choose to collect statistics on columns that make up all the indexes defined on the table. It is assumed here that critical columns in queries are also those used to create indexes on the table. If there are no indexes on the table, it is as good as an empty list and no column statistics will be collected. It can be used in the on-cols-clause or the on-dist-cols-clause. It is redundant in the on-cols-clause if specified in both clauses since the WITH DISTRIBUTION clause is used to specify collection of both basic and distribution statistics. XML type columns are by definition not a key column and will not be included for statistics collection by the ON KEY COLUMNS clause. This option cannot be used 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 non-existent 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 will be collected.

If the column appears in both lists, distribution statistics will be collected (unless NUM_FREQVALUES and NUM_QUANTILES are set to zero).

NUM_FREQVALUES

Defines the maximum number of frequency values to collect. It can be specified for an individual column in the ON COLUMNS clause. If the value is not specified for an individual column, the frequency limit value will be picked up from that specified in the DEFAULT clause. If it is not specified there either, the maximum number of frequency values to be collected will be what is set in 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 not specified for an individual column, the quantile limit value will be picked up from that specified in the DEFAULT clause. If it is not specified there either, the maximum number of quantile values to be collected will be what is set in the NUM_QUANTILES database configuration parameter.

WITH DISTRIBUTION

This clause specifies that both basic statistics and distribution statistics are to be collected on the columns. If the ON COLUMNS clause is not specified, distribution statistics are collected on all the columns of the table or statistical view (excluding columns that are ineligible such as CLOB and LONG VARCHAR). If the ON COLUMNS clause is specified, distribution statistics are collected only on the column list provided (excluding those ineligible for statistics collection). If the clause is not specified, only basic statistics are collected.

Collection of distribution statistics on column groups is currently not supported; distribution statistics will not be collected when column groups are specified in the WITH DISTRIBUTION ON COLUMNS clause.

DEFAULT

If NUM_FREQVALUES or NUM_QUANTILES are specified, these values will be used to determine the maximum number of frequency and quantile statistics to be collected for the columns, if these are not specified for individual columns in the ON COLUMNS clause. If the DEFAULT clause is not specified, the values used will be those in the corresponding database configuration parameters.

LIKE STATISTICS

When this option is specified additional column statistics are collected. These statistics are the SUB_COUNT and the SUB_DELIM_LENGTH statistics in SYSSTAT.COLUMNS. They are collected for string columns only and they are used by the query optimizer to improve the selectivity estimates for predicates of the type "column LIKE '%xyz'" and "column LIKE '%xyz%'"

ALLOW WRITE ACCESS

Specifies that other users can read from and write to the table(s) while statistics are calculated. For statistical views, these are the base tables referenced in the view definition.

The ALLOW WRITE ACCESS option is not recommended for tables that will have a lot of inserts, updates or deletes occurring concurrently. The RUNSTATS command first performs 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 should 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 table(s) while statistics are calculated. For statistical views, these are the base tables referenced in the view definition.

TABLESAMPLE BERNOULLI

This option allows **RUNSTATS** to collect statistics on a sample of the rows from the table or statistical view. Bernoulli sampling considers each row individually, including that row with probability $P/100$ (where P is the value of numeric-literal) and excluding it with probability $1-P/100$. Thus, if the numeric-literal were evaluated to be the value 10, representing a 10 percent sample, each row would be included with probability 0.1 and be excluded with probability 0.9. Unless the optional **REPEATABLE** clause is specified, each execution of **RUNSTATS** will usually yield a different such sample of the table. All data pages will be retrieved through a table scan but only the percentage of rows as specified through the numeric-literal parameter will be used for the statistics collection.

TABLESAMPLE SYSTEM

This option allows **RUNSTATS** to collect statistics on a sample of the data pages from the table(s). System sampling considers each page individually, including that page with probability $P/100$ (where P is the value of numeric-literal) and excluding it with probability $1-P/100$. Unless the optional **REPEATABLE** clause is specified, each execution of **RUNSTATS** will usually yield a different such sample of the table. The size of the sample is controlled by the numeric-literal parameter in parentheses, representing an approximate percentage P of the table to be returned. Only a percentage of the data pages as specified through the numeric-literal parameter will be retrieved and used for the statistics collection. On statistical views, system sampling is restricted to a specific class of views. These are views that either access a single base table or nickname, or that access multiple bases tables that are joined via referential-integrity relationships. In either case, there must not be any local predicates in the view definition. If system sampling is specified on a view that cannot support such sampling, an SQL20288N error is raised.

REPEATABLE (*integer-literal*)

Adding the **REPEATABLE** clause to the **TABLESAMPLE** clause ensures that repeated executions of **RUNSTATS** return the same sample. The *integer-literal* parameter is a non-negative integer representing the seed to be used in sampling. Passing a negative seed will result in an error (SQL1197N). The sample set might still vary between repeatable **RUNSTATS** invocations if activity against the table or statistical view resulted in changes to the table or statistical view data since the last time **TABLESAMPLE REPEATABLE** was run. Also, the method by which the sample was obtained as specified by the *bernoulli* or *system* keyword, must also be the same to ensure consistent results.

numeric-literal

The numeric-literal parameter 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 between 1 and 0. For example, a value of 0.01 represents one one-hundredth of a percent, such that 1 row in 10,000 would be sampled, on average. A value of 0 or 100 will be treated by the DB2 database system as if sampling was not specified, regardless of

whether TABLESAMPLE BERNOULLI or TABLESAMPLE SYSTEM is specified. A value greater than 100 or less than 0 will be treated by DB2 as an error (SQL1197N).

SET PROFILE NONE

Specifies that no statistics profile will be set for this **RUNSTATS** invocation.

SET PROFILE

Allows **RUNSTATS** to generate and store a specific statistics profile in the system catalog tables and executes the **RUNSTATS** command options to gather statistics.

SET PROFILE ONLY

Allows **RUNSTATS** to generate and store a specific statistics profile in the system catalog tables without running the **RUNSTATS** command options.

UPDATE PROFILE

Allows **RUNSTATS** to modify an existing statistics profile in the system catalog tables, and runs the **RUNSTATS command options of the updated statistics profile to gather statistics**.

UPDATE PROFILE ONLY

Allows **RUNSTATS** to modify an existing statistics profile in the system catalog tables without running the **RUNSTATS** command options of the updated statistics profile.

UTIL_IMPACT_PRIORITY *priority*

Specifies that **RUNSTATS** will be throttled at the level 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 those at higher priorities. If *priority* is not specified, the **RUNSTATS** will have the default priority of 50. Omitting the **UTIL_IMPACT_PRIORITY** keyword will invoke 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 will run unthrottled. This option cannot be used for views.

In a partitioned database, when used on tables, the **RUNSTATS** command collects the statistics on only a single database partition. If the database partition from which the **RUNSTATS** command is executed has a partition of the table, then the command executes on that database partition. Otherwise, the command executes on the first database partition in the database partition group across which the table is partitioned.

Usage Notes:

1. When there are detached partitions on a partitioned table, index keys that still belong to detached data partitions which require cleanup will not be 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 collected before asynchronous index cleanup is run will be misleading. If the **RUNSTATS** command is issued before asynchronous index cleanup completes, it will likely generate a false alarm for index reorganization or index cleanup based on the inaccurate statistics. Once asynchronous index

cleanup is run, all the index keys that still belong to detached data partitions which require cleanup will be removed and this may eliminate the need for index reorganization.

For partitioned tables, you are encouraged to issue the **RUNSTATS** command after an asynchronous index cleanup has completed in order to generate accurate index statistics in the presence of detached data partitions. To determine whether or not there are detached data partitions in the table, you can check the status field in the `SYSDATAPARTITIONS` table and look for the value I (index cleanup) or D (detached with dependant MQT).

2. Command execution status is returned in the `SQLCA` resulting from the `CALL` statement.
3. It is recommended to run the **RUNSTATS** command:
 - On tables that have been modified considerably (for example, if a large number of updates have been made, or if a significant amount of data has been inserted or deleted or if **LOAD** has been done without the statistics option during **LOAD**).
 - On tables that have been reorganized (using **REORG**, **REDISTRIBUTE DATABASE PARTITION GROUP**).
 - On tables which have been row compressed.
 - When a new index has been created.
 - Before binding applications whose performance is critical.
 - When the prefetch quantity is changed.
 - On statistical views whose underlying tables have been modified substantially so as to change the rows that are returned by the view.
 - After **LOAD** has been executed with the `STATISTICS` option, use the **RUNSTATS** utility to collect statistics on XML columns. Statistics for XML columns are never collected during **LOAD**, even when **LOAD** is executed with the `STATISTICS` option. When **RUNSTATS** is used to collect statistics for XML columns only, existing statistics for non-XML columns that have been collected by **LOAD** or a previous execution of the **RUNSTATS** utility are retained. In the case where statistics on some XML columns have been collected previously, the previously collected statistics for an XML column will either be dropped if no statistics on that XML column are collected by the current command, or be replaced if statistics on that XML column are collected by the current command.
4. The options chosen must depend on the specific table and the application. In general:
 - If the table is a very 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 execute **RUNSTATS** limited to the set of columns that are used in predicates. This way, you will be able to execute the **RUNSTATS** command more often.
 - If time to collect statistics is very 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 very limited and table statistics are to be gathered, consider using the TABLESAMPLE option to collect statistics on a subset of the table data.
 - If there are many indexes on the table and DETAILED (extended) information on the indexes might improve access plans, consider the SAMPLED option to reduce the time it takes to collect statistics. Regardless of whether you use the SAMPLED option, collecting detailed statistics on indexes is time consuming. Do not collect these statistics unless you are sure that they will be useful for your queries.
 - If there is skew in certain columns and predicates of the type "column = constant", it might be beneficial to specify a larger NUM_FREQVALUES value for that column
 - Collect distribution statistics for all columns that are used in equality predicates and for which the distribution of values might be skewed.
 - For columns that have range predicates (for example "column >= constant", "column BETWEEN constant1 AND constant2") or of the type "column LIKE '%xyz'", it might be beneficial to specify a larger NUM_QUANTILES value.
 - If storage space is a concern and one cannot afford too much time on collecting statistics, do not specify high NUM_FREQVALUES or NUM_QUANTILES values for columns that are not used in predicates.
 - If index statistics are requested, and statistics have never been run on the table containing the index, statistics on both the table and indexes are calculated.
 - If statistics for XML columns in the table are not required, the EXCLUDING XML COLUMNS option can be used to exclude all XML columns. This option takes precedence over all other clauses that specify XML columns for statistics collection.
5. After the command is run note the following:
 - A COMMIT should be issued to release the locks.
 - To allow new access plans to be generated, the packages that reference the target table must be rebound.
 - Executing the command on portions of the table could result in inconsistencies as a result of activity on the table since the command was last issued. In this case a warning message is returned. Issuing **RUNSTATS** on the table only might make table 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 **RUNSTATS** on the table only, the table cardinality might be less than FIRSTKEYCARD, which is an inconsistency. In the same way, if you collect statistics on a new index when you create it, the table level statistics might be inconsistent.
 6. The **RUNSTATS** command will drop previously collected distribution statistics if table statistics are requested. For example, **RUNSTATS ON TABLE**, or **RUNSTATS ON TABLE ... AND INDEXES ALL** will cause previously collected distribution statistics to be dropped. If the command is run on indexes only then previously collected distribution statistics are retained. For example, **RUNSTATS ON TABLE ... FOR INDEXES ALL** will cause the previously collected distribution statistics to be retained. If the **RUNSTATS** command is run on XML columns only, then previously collected basic column statistics and distribution statistics are retained. In the case where statistics on some XML columns have been collected previously, the previously collected statistics for an XML column will either be dropped if no statistics on that XML column

RUNSTATS using ADMIN_CMD

are collected by the current command, or be replaced if statistics on that XML column are collected by the current command.

7. For range-clustered tables, there is a special system-generated index in the catalog tables which represents the range ordering property of range-clustered tables. When statistics are collected on this type of table, if the table is to be included as part of the statistics collection, statistics will also be 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 having the base data clustered perfectly along the index order.
8. In the on-dist-cols clause of the command syntax, the Frequency Option and Quantile Option parameters are currently not supported for Column GROUPS. These options are supported for single columns.
9. There are three prefetch statistics that cannot be computed when working in DMS mode. When looking at 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
10. Runstats sampling through TABLESAMPLE only occurs with table data pages and not index pages. When index statistics as well as sampling is requested, all the index pages are scanned for statistics collection. It is only in the collection of table statistics where TABLESAMPLE is applicable. However, a more efficient collection of detailed index statistics is available through the SAMPLED DETAILED option. This is a different method of sampling than that employed by TABLESAMPLE and only applies to the detailed set of index statistics.
11. A statistics profile can be set or updated for the table or statistical view specified in the **RUNSTATS** command, by using the set profile or update profile options. The statistics profile is stored in a visible string format, which represents the **RUNSTATS** command, in the STATISTICS_PROFILE column of the SYSIBM.SYSTABLES system catalog table.
12. Statistics collection on XML type columns is governed by two DB2 database system registry values: DB2_XML_RUNSTATS_PATHID_K and DB2_XML_RUNSTATS_PATHVALUE_K. These two parameters are similar to the NUM_FREQVALUES parameter in that they specify the number of frequency values to collect. If not set, a default of 200 will be used for both parameters.
13. **RUNSTATS** acquires an IX table lock on SYSTABLES and a U lock on the row for the table on which stats are being gathered at the beginning of **RUNSTATS**. Operations can still read from SYSTABLES including the row with the U lock. Write operations are also possible, providing they do not occur against the row with the U lock. However, another reader or writer will not be able acquire an S lock on SYSTABLES because of **RUNSTATS'** IX lock.

Example:

Collect statistics on all columns used in indexes and on all indexes.

```
CALL SYSPROC.ADMIN_CMD ('RUNSTATS ON TABLE db2user.employee  
ON KEY COLUMNS and INDEXES ALL')
```

Related concepts:

- “Automatic statistics collection” in *Performance Guide*

Related tasks:

- “Collecting catalog statistics” in *Performance Guide*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “REORG INDEXES/TABLE command using the ADMIN_CMD procedure” on page 126
- “ADMIN_COPY_SCHEMA procedure – Copy a specific schema and its objects” on page 498
- “db2Runstats API - Update statistics for tables and indexes” in *Administrative API Reference*

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:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required connection:

Database.

Command syntax:

```
▶▶ SET TAPE POSITION ON device TO position ▶▶
```

Command parameters:

ON device

Specifies a valid tape device name. The default value is `\\.\TAPE0`. The device specified must be relative to the server.

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.

Example:

Because DB2 writes a tape mark after every backup image, specifying a position of 1 will move the tape to the start of the second archive on the tape.

```
CALL SYSPROC.ADMIN_CMD( 'set tape position to 1' )
```

Usage note:

Command execution status is returned in the SQLCA resulting from the CALL statement.

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “INITIALIZE TAPE command using the ADMIN_CMD procedure” on page 94
- “REWIND TAPE command using the ADMIN_CMD procedure” on page 143

UNQUIESCE DATABASE command using the ADMIN_CMD procedure

Restores user access to databases which have been quiesced for maintenance or other reasons. UNQUIESCE restores user access without necessitating a shutdown and database restart.

Unless specifically designated, no user except those with *sysadm*, *sysmaint*, or *sysctrl* has access to a database while it is quiesced. Therefore an UNQUIESCE is required to restore general access to a quiesced database.

Scope:

UNQUIESCE DB restores user access to all objects in the quiesced database.

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:

For database level unquiesce:

- *sysadm*
- *dbadm*

Command syntax:

►►—UNQUIESCE—DB—►►

Required connection:

Database

Command parameters:

DB Unquiesce the database. User access will be restored to all objects in the database.

Examples:

Unquiescing a Database

```
CALL SYSPROC.ADMIN_CMD( 'unquiesce db' )
```

This command will unquiesce the database that had previously been quiesced.

Usage note:

Command execution status is returned in the SQLCA resulting from the CALL statement.

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “QUIESCE DATABASE command using the ADMIN_CMD procedure” on page 117

UNQUIESCE DATABASE using ADMIN_CMD

- “QUIESCE TABLESPACES FOR TABLE command using the ADMIN_CMD procedure” on page 119
- “db2DatabaseUnquiesce API - Unquiesce database” in *Administrative API Reference*

UPDATE ALERT CONFIGURATION command using the ADMIN_CMD procedure

Updates the alert configuration settings for health indicators.

Authorization:

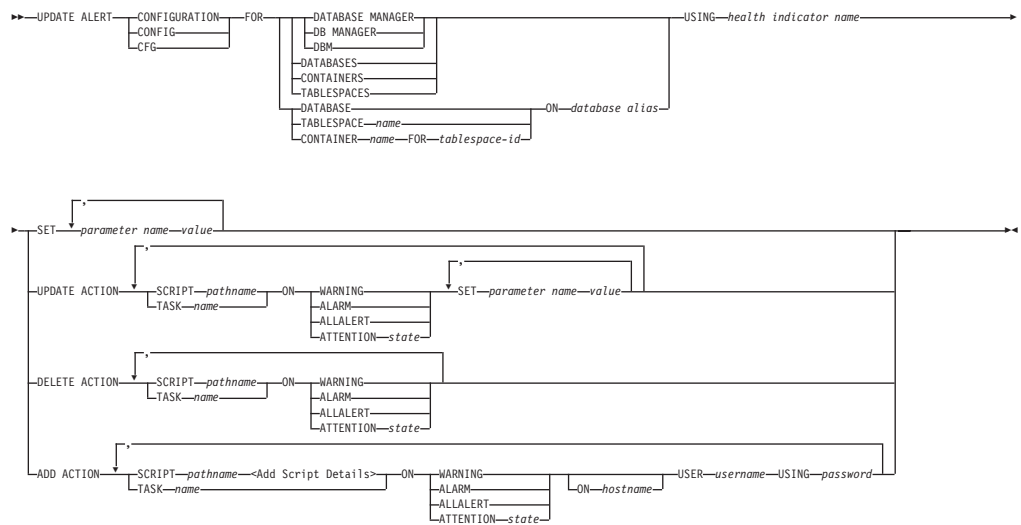
One of the following:

- sysadm
- sysmaint
- sysctrl

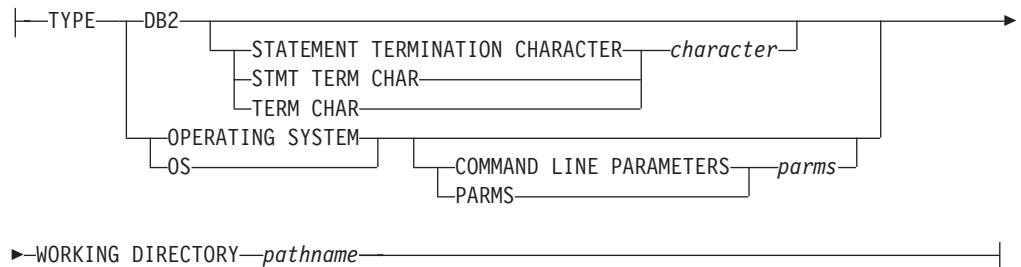
Required Connection:

Database.

Command Syntax:



Add Script Details:



Command Parameters:

DATABASE MANAGER

Updates alert settings for the database manager.

DATABASES

Updates alert settings for all databases managed by the database manager.

UPDATE ALERT CONFIGURATION using ADMIN_CMD

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 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 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 *name* FOR *tablespace-id* ON *database alias*

Updates the alert settings for the table space container called *name*, for the table space specified using the "FOR tablespace-id" 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 *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:

- 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.

The list of possible health indicator units for your specific DB2 version can be gathered by running the following query :

```
SELECT SUBSTR(UNIT,1,80) AS UNIT  
FROM TABLE(HEALTH_GET_IND_DEFINITION('')) AS T GROUP BY UNIT
```

UPDATE ALERT CONFIGURATION using ADMIN_CMD

UPDATE ACTION SCRIPT *pathname* ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]

Specifies that the script attributes of the predefined script with absolute pathname *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:

- 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 *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:

- USERID
- PASSWORD
- SYSTEM

DELETE ACTION SCRIPT *pathname* ON [WARNING | ALARM | ALLALERT | ATTENTION *state*]

Removes the action script with absolute pathname *pathname* from the list of alert action scripts.

DELETE ACTION 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 pathname *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

UPDATE ALERT CONFIGURATION using ADMIN_CMD

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 pathname, *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. When using the `ADD ACTION` option, the *username* and *password* might be exposed in the network (where the *username* and *password* are sent unencrypted), the `db2diag.log`, trace files, dump file, snapshot monitor (dynamic SQL snapshot), system monitor snapshots, a number of event monitors (such as statement, deadlock), Query Patroller, explain tables, `db2pd` output (such as package cache and lock timeout mechanisms) and `db2` audit records.

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 | ATTENTION] *state*

Specifies the condition on which the action will run. For threshold-based HIs, this is WARNING or ALARM. For state-based HIs, this can be a numeric state as documented in a table to be provided for each state-based HI (for example, table space states), or a text identifier for this state.

Example:

Add an action for the `db.log_fs_util` indicator that will execute the script `/home/test/scripts/logfsutilact` when there is an alarm on the system with hostname 'plato'.

```
CALL SYSPROC.ADMIN_CMD( 'update alert cfg for databases using
db.log_fs_util add action script /home/test/scripts/logfsutilact
type os command line parameters "param1 param2" working
directory /tmp on alarm on plato user dricard using mypasswdv' )
```

To check the alert configuration after it has been set, you can use the `HEALTH_GET_IND_DEFINITION` and `HEALTH_GET_ALERT_ACTION_CFG` table functions as follows:

```
SELECT OBJECTTYPE, ID, CONDITION, ACTIONTYPE,
SUBSTR(ACTIONNAME,1,50) AS ACTION_NAME
FROM TABLE(SYSPROC.HEALTH_GET_ALERT_ACTION_CFG('DB','G','',''))
AS ALERT_ACTION_CFG
```

The following is an example of output from this query:

UPDATE ALERT CONFIGURATION using ADMIN_CMD

| OBJECTTYPE | ID | CONDITION | ACTIONTYPE | ACTION_NAME |
|------------|------|-----------|------------|------------------------------------|
| DB | 1006 | ALARM | S | /home/dricard/scripts/logfsutilact |

1 record(s) selected. ...

Usage notes:

Command execution status is returned in the SQLCA resulting from the CALL statement.

The *database alias* must be defined in the catalog on the server and be local to the server.

The *pathname* must be with a fully-qualified server path name.

Related tasks:

- “Configuring health indicators using a client application” in *System Monitor Guide and Reference*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “RESET ALERT CONFIGURATION command using the ADMIN_CMD procedure” on page 136
- “db2UpdateAlertCfg API - Update the alert configuration settings for health indicators” in *Administrative API Reference*

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:

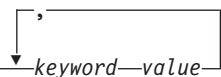
None.

Required connection:

Database. The DAS must be running.

Command syntax:

```
►► UPDATE CONTACT name USING keyword value ◄◄
```



Command parameters:

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.

Example:

Update the address of user 'test' to 'newaddress@test.com'.

```
CALL SYSPROC.ADMIN_CMD( 'update contact test using address newaddress@test.com' )
```

Usage notes:

The DAS must have been created and be running.

Command execution status is returned in the SQLCA resulting from the CALL statement.

Related reference:

UPDATE CONTACT using ADMIN_CMD

- “ADMIN_CMD – Run administrative commands” on page 38
- “ADD CONTACT command using the ADMIN_CMD procedure” on page 44
- “ADD CONTACTGROUP command using the ADMIN_CMD procedure” on page 46
- “DROP CONTACT command using the ADMIN_CMD procedure” on page 68
- “DROP CONTACTGROUP command using the ADMIN_CMD procedure” on page 69
- “UPDATE CONTACTGROUP command using the ADMIN_CMD procedure” on page 166
- “db2admin - DB2 administration server command” in *Command Reference*
- “db2UpdateContact API - Update the attributes of a contact” in *Administrative API Reference*

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:

Database. The DAS must be running.

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.

Example:

Add the contact named 'cname2' to the contact group named 'gname1':

```
CALL SYSPROC.ADMIN_CMD( 'update contactgroup gname1 add contact cname2' )
```

Usage notes:

The DAS must have been created and be running.

UPDATE CONTACTGROUP using ADMIN_CMD

Command execution status is returned in the SQLCA resulting from the CALL statement.

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “ADD CONTACT command using the ADMIN_CMD procedure” on page 44
- “ADD CONTACTGROUP command using the ADMIN_CMD procedure” on page 46
- “DROP CONTACT command using the ADMIN_CMD procedure” on page 68
- “DROP CONTACTGROUP command using the ADMIN_CMD procedure” on page 69
- “UPDATE CONTACT command using the ADMIN_CMD procedure” on page 164
- “db2admin - DB2 administration server command” in *Command Reference*
- “db2UpdateContactGroup API - Update the attributes of a contact group” in *Administrative API Reference*

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 only affects the database partition on which it is executed.

Authorization:

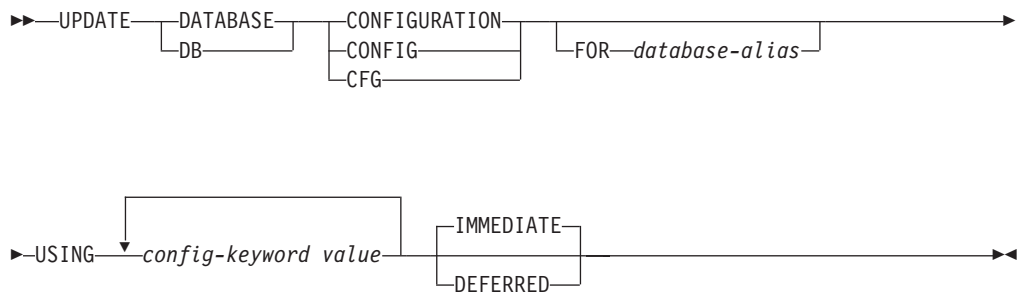
One of the following:

- *sysadm*
- *sysctrl*
- *sysmaint*

Required connection:

Database. The database connection must be local to the instance containing the connected database.

Command syntax:



Command parameters:

DEFERRED

Make the changes only in the configuration file, so that the changes take effect the next time you reactivate the database.

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. The database alias must be defined locally on the server. 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.

UPDATE DATABASE CONFIGURATION using ADMIN_CMD

- 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.

IMMEDIATE

Make the changes immediately, while the database is running. IMMEDIATE is the default action. Since the ADMIN_CMD procedure requires a database connection, the changes will be effective immediately for any dynamically configurable parameters for the connected database.

USING *config-keyword value*

config-keyword specifies the database configuration parameter to be updated. *value* specifies the value to be assigned to the parameter.

Example:

Set the database configuration parameter *sortheap* to a value of 1000 on the database partition to which the application is currently connected to.

```
CALL SYSPROC.ADMIN_CMD ('UPDATE DB CFG USING sortheap 1000')
```

Usage notes:

Command execution status is returned in the SQLCA resulting from the CALL statement.

The *database-alias* must be an alias name that is defined on the server.

The command affects only the database partition to which the application is currently connected.

To view or print a list of the database configuration parameters, use the SYSIBMADM.DBCFG administration view.

To reset all the database configuration parameters to the recommended defaults, use the RESET DATABASE CONFIGURATION command using the ADMIN_CMD procedure.

To change a database configuration parameter, use the UPDATE DATABASE CONFIGURATION command through the ADMIN_CMD procedure. For example, to change the logging mode to “archival logging” on a single-partition database environment containing a database called ZELLMART, use:

```
CALL SYSPROC.ADMIN_CMD ('update db cfg for zellmart using logretain recovery')
```

To check that the *logretain* configuration parameter has changed, use:

```
SELECT * FROM SYSIBMADM.DBCFG WHERE NAME='logretain'
```

It is recommended that the database configuration parameters be set to the same value on all database partitions. You can do this for each database partition by:

1. setting the DB2NODE variable to a database partition number
2. connecting to the database partition
3. updating the database configuration parameters using UPDATE DATABASE CONFIGURATION command through the ADMIN_CMD procedure
4. disconnecting from the database partition

UPDATE DATABASE CONFIGURATION using ADMIN_CMD

For more information about DB2 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 on which parameters are configurable on-line and which ones are not, see summary list of configuration parameters.

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.

Related tasks:

- “Configuring DB2 with configuration parameters” in *Performance Guide*

Related reference:

- “GET DATABASE CONFIGURATION command” in *Command Reference*
- “db2CfgSet API - Set the database manager or database configuration parameters” in *Administrative API Reference*
- “Configuration parameters summary” in *Performance Guide*
- “ADMIN_CMD – Run administrative commands” on page 38
- “DBCFCG administrative view – Retrieve database configuration parameter information” on page 182
- “RESET DATABASE CONFIGURATION command using the ADMIN_CMD procedure” on page 139
- “RESET DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure” on page 141
- “UPDATE DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure” on page 171

UPDATE DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure

Modifies individual entries in the database manager configuration file for the instance that contains the currently connected database.

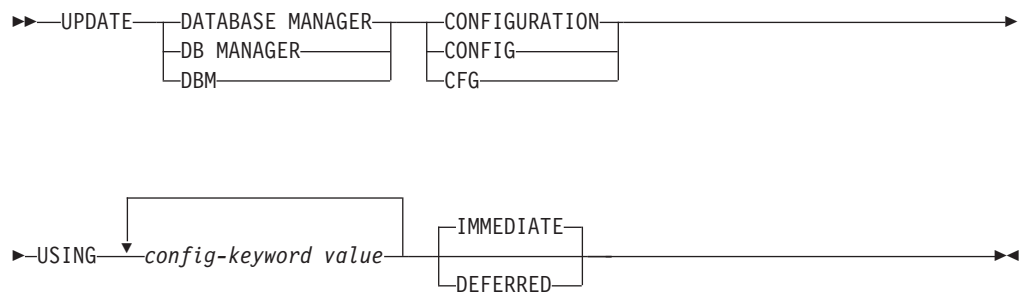
Authorization:

sysadm

Required connection:

Database.

Command syntax:



Command parameters:

DEFERRED

Make the changes only in the configuration file, so that the changes take effect when the instance is restarted.

IMMEDIATE

Make the changes right now, dynamically, while the instance is running. IMMEDIATE is the default. If a parameter supports dynamic update, an attempt is made to update the parameter dynamically and the authorization used is the user ID that established the connection.

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.

Example:

Update the diagnostic level to 1 for the database manager configuration.

```
CALL SYSPROC.ADMIN_CMD('db2 update dbm cfg using DIAGLEVEL 1')
```

Usage notes:

To view or print a list of the database manager configuration parameters, use the SYSIBMADM.DBMCFG administrative view. To reset the database manager configuration parameters to the recommended database manager defaults, use the RESET DATABASE MANAGER CONFIGURATION command through the ADMIN_CMD procedure. For more information about database manager configuration parameters and the values of these parameters appropriate for each

UPDATE DATABASE MANAGER CONFIGURATION using ADMIN_CMD

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 on which parameters are configurable on-line 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 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 TPNNAME 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.

Command execution status is returned in the SQLCA resulting from the CALL statement.

Updates can only be made to the database instance that contains the connected database.

If a parameter supports dynamic update, an attempt is made to update it dynamically, even if the IMMEDIATE keyword is not specified. The authorization used is the current SYSTEM_USER id.

Related tasks:

- “Configuring DB2 with configuration parameters” in *Performance Guide*

Related reference:

- “GET DATABASE MANAGER CONFIGURATION command” in *Command Reference*
- “TERMINATE command” in *Command Reference*
- “Configuration parameters summary” in *Performance Guide*
- “db2CfgSet API - Set the database manager or database configuration parameters” in *Administrative API Reference*
- “ADMIN_CMD – Run administrative commands” on page 38
- “DBMCFG administrative view – Retrieve database manager configuration parameter information” on page 184
- “RESET DATABASE CONFIGURATION command using the ADMIN_CMD procedure” on page 139
- “RESET DATABASE MANAGER CONFIGURATION command using the ADMIN_CMD procedure” on page 141

UPDATE DATABASE MANAGER CONFIGURATION using ADMIN_CMD

- “UPDATE DATABASE CONFIGURATION command using the ADMIN_CMD procedure” on page 168

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:

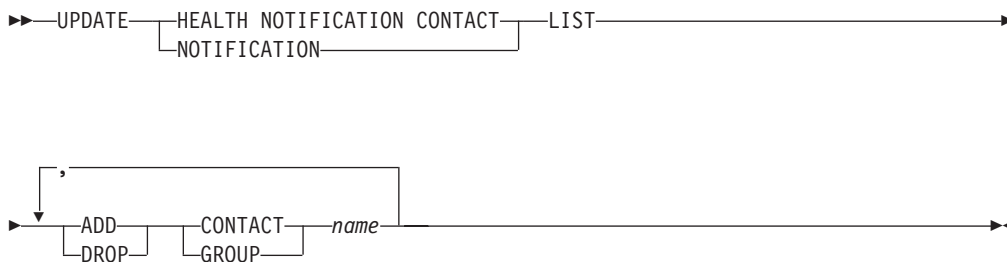
One of the following:

- sysadm
- sysctrl
- sysmaint

Required Connection:

Database.

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.

Example:

Add the contact group 'gname1' to the health notification contact list:

```
CALL SYSPROC.ADMIN_CMD( 'update notification list add group gname1' )
```

Usage note:

Command execution status is returned in the SQLCA resulting from the CALL statement.

Related tasks:

- "Enabling health alert notification" in *System Monitor Guide and Reference*

UPDATE HEALTH NOTIFICATION CONTACT LIST using ADMIN_CMD

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38
- “db2UpdateHealthNotificationList API - Update the list of contacts to whom health alert notifications can be sent” in *Administrative API Reference*

UPDATE HISTORY command using the ADMIN_CMD procedure

Updates the location, device type, comment, or status in a history file entry on the currently connected database partition.

Authorization:

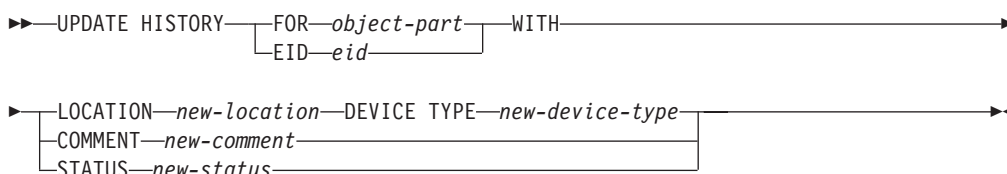
One of the following:

- *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:

- | | |
|----------|-------------|
| D | Disk |
| K | Diskette |
| T | Tape |
| A | TSM |
| U | User exit |
| P | Pipe |
| N | Null device |
| X | XBSA |

Q SQL statement

O Other

COMMENT *new-comment*

Specifies a new comment to describe the entry.

STATUS *new-status*

Specifies a new status for an entry. Only backup entries can have their status updated. Valid values are:

A Active. Most entries are active.

I Inactive. Backup images that are no longer on the active log chain become 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.

Example:

To update the history file entry for a full database backup taken on April 13, 1997 at 10:00 a.m., enter:

```
CALL SYSPROC.ADMIN_CMD('update history
  for 19970413100000001 with location
  /backup/dbbackup.1 device type d')
```

Usage notes:

The primary purpose of the database history file 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.

Command execution status is returned in the SQLCA resulting from the CALL statement.

The *object-part* or *eid* must refer to the log history entries on the connected database partition.

Related concepts:

- “Developing a backup and recovery strategy” in *Data Recovery and High Availability Guide and Reference*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38

UPDATE STMM TUNING DBPARTITIONNUM command using the ADMIN_CMD procedure

Update the user preferred self tuning memory manager (STMM) tuning database partition.

Authorization:

SYSADM or DBADM authority

Required connection:

Database

Command syntax:

►►—UPDATE—STMM—TUNING—DBPARTITIONNUM—*partitionnum*—◄◄

Command parameter:

partitionnum

partitionnum is an integer. If -1 or a non-existing database partition number is used, the STMM tuning database partition will use the default database partition as defined for the STMM feature. If -1 or a non-existing database partition number is used, DB2 will automatically select an appropriate database partition on which to run the STMM memory tuner.

Example:

Update the user preferred self tuning memory manager (STMM) tuning database partition to database partition 3.

```
CALL SYSPROC.ADMIN_CMD( 'update stmm tuning dbpartitionnum 3' )
```

Usage notes:

The STMM tuning process periodically checks for a change in the user preferred STMM tuning database partition number value. The STMM tuning process will move to the user preferred STMM tuning database partition if *partitionnum* exists and is an active database partition. Once this command changes the STMM tuning database partition number an immediate change is made to the current STMM tuning database partition number.

Command execution status is returned in the SQLCA resulting from the CALL statement.

This command commits its changes in the ADMIN_CMD procedure.

Related concepts:

- “Using self tuning memory in partitioned database environments” in *Performance Guide*
- “Self tuning memory” in *Performance Guide*

Related reference:

- “ADMIN_CMD – Run administrative commands” on page 38

UPDATE STMM TUNING DBPARTITIONNUM using ADMIN_CMD

- “GET STMM TUNING DBPARTITIONNUM command using the ADMIN_CMD procedure” on page 78

Configuration administrative SQL routines and views

DB_PARTITIONS

The DB_PARTITIONS table function returns the contents of the db2nodes.cfg file in table form.

Syntax:

▶▶—DB_PARTITIONS—(—)—▶▶

The schema is SYSPROC.

Authorization:

EXECUTE privilege on the DB_PARTITIONS table function.

Table function parameters:

The function has no input parameters.

Example:

Retrieve information from a 3 logical partition database.

```
SELECT * FROM TABLE(DB_PARTITIONS()) AS T
```

The following is an example of output from this query.

```
PARTITION_NUMBER HOST_NAME          PORT_NUMBER SWITCH_NAME
-----
0 jessicae.torolab.ibm.com          0 jessicae
1 jessicae.torolab.ibm.com          1 jessicae
2 jessicae.torolab.ibm.com          2 jessicae
```

3 record(s) selected.

Information returned:

Table 38. Information returned by the DB_PARTITIONS table function

| Column name | Data type | Description |
|------------------|--------------|--|
| PARTITION_NUMBER | SMALLINT | A unique number between 0 and 999 that identifies a database partition server in a partitioned database environment. |
| HOST_NAME | VARCHAR(128) | The TCP/IP host name of the database partition server. |
| 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. |

Related reference:

- “Supported administrative SQL routines and views” on page 8

DBCFCG administrative view – Retrieve database configuration parameter information

The DBCFCG administrative view retrieves database configuration parameter information for the currently connected database for all database partitions.

The schema is SYSIBMADM.

Authorization:

SELECT or CONTROL privilege on the DBCFCG administrative view and EXECUTE privilege on the DB_GET_CFG table function.

Examples:

Example 1: Retrieve the automatic maintenance settings in the database configuration that are stored in memory for all database partitions.

```
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_tbt_maint | OFF |
| 0 | auto_runstats | OFF |
| 0 | auto_stats_prof | OFF |
| 0 | auto_prof_upd | OFF |
| 0 | auto_reorg | OFF |
| 0 | autorestart | ON |

8 record(s) selected.

Example 2: Retrieve all the database configuration parameters values stored on disk for all database partitions.

```
SELECT NAME, DEFERRED_VALUE, DBPARTITIONNUM FROM SYSIBMADM.DBCFCG
```

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 |
| ... | | |

Information returned:

Table 39. Information returned by the DBCFCG administrative view

| Column name | Data type | Description |
|-------------|-------------|-------------------------------|
| NAME | VARCHAR(32) | Configuration parameter name. |

Table 39. Information returned by the DBCFCG administrative view (continued)

| Column name | Data type | Description |
|----------------------|---------------|--|
| VALUE | VARCHAR(1024) | The current value of the configuration parameter stored in memory. |
| VALUE_FLAGS | VARCHAR(10) | Provides specific information for the configuration parameter current value. Valid values are: <ul style="list-style-type: none"> • NONE - no additional information • AUTOMATIC - the configuration parameter has been set to automatic |
| 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"> • NONE - no additional information • AUTOMATIC - the configuration parameter has been set to automatic |
| DATATYPE | VARCHAR(128) | Configuration parameter data type. |
| DBPARTITIONNUM | SMALLINT | Database partition number. |

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Configuration parameters summary” in *Performance Guide*
- “Administrative views versus table functions” on page 3
- “Authorization for administrative views” on page 6
- “DBMCFG administrative view – Retrieve database manager configuration parameter information” on page 184

DBMCFG administrative view – Retrieve database manager configuration parameter information

The DBMCFG administrative view returns database manager configuration parameter information including the values in memory and the values stored on disk.

The schema is SYSIBMADM.

Authorization:

SELECT or CONTROL privilege on the DBMCFG administrative view and EXECUTE privilege on the DBM_GET_CFG table function.

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 |
| aslheapsz | 15 |
| audit_buf_sz | 0 |
| authentication | SERVER |
| catalog_noauth | YES |
| clnt_krb_plugin | |
| ... | |
| comm_bandwidth | 0.000000e+00 |
| conn_elapse | 0 |
| cpuspeed | 4.000000e-05 |
| dft_account_str | |
| dft_mon_bufpool | OFF |
| ... | |
| dft_mon_timestamp | ON |
| dft_mon_uow | OFF |
| ... | |
| jdk_path | /wsdb/v91/b1dsupp/AIX5L |
| ... | |

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 | ... |
| aslheapsz | 15 | NONE | ... |
| audit_buf_sz | 0 | NONE | ... |
| authentication | SERVER | NONE | ... |
| catalog_noauth | YES | NONE | ... |
| clnt_krb_plugin | | NONE | ... |
| clnt_pw_plugin | | NONE | ... |
| comm_bandwidth | 0.000000e+00 | NONE | ... |
| conn_elapse | 0 | NONE | ... |
| cpuspeed | 4.000000e-05 | 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      ...
...
dir_cache               YES       NONE      ...
discover                SEARCH    NONE      ...
discover_inst           ENABLE    NONE      ...
fcm_num_anchors         0        AUTOMATIC ...
fcm_num_buffers         0        AUTOMATIC ...
fcm_num_connect         0        AUTOMATIC ...
...

```

Output for this query (continued).

```

... DEFERRED_VALUE      DEFERRED_VALUE_FLAGS  DATATYPE
... -----
... 0                   NONE                   INTEGER
... -1                  NONE                   INTEGER
... 15                  NONE                   BIGINT
... 0                   NONE                   BIGINT
... SERVER              NONE                   VARCHAR(32)
... YES                 NONE                   VARCHAR(3)
...                    NONE                   VARCHAR(32)
...                    NONE                   VARCHAR(32)
... 0.000000e+00       NONE                   REAL
... 0                   NONE                   INTEGER
... 4.000000e-05       NONE                   REAL
...                    NONE                   VARCHAR(25)
... OFF                 NONE                   VARCHAR(3)
... OFF                 NONE                   VARCHAR(3)
... OFF                 NONE                   VARCHAR(3)
... OFF                 NONE                   VARCHAR(3)
... OFF                 NONE                   VARCHAR(3)
...
... YES                 NONE                   VARCHAR(3)
... SEARCH              NONE                   VARCHAR(8)
... ENABLE              NONE                   VARCHAR(8)
... 0                   AUTOMATIC              BIGINT
... 512                 AUTOMATIC              BIGINT
... 0                   AUTOMATIC              BIGINT
...

```

Information returned:

Table 40. Information returned by the DBMCFG administrative view

| Column name | Data type | Description |
|-------------|--------------|--|
| NAME | VARCHAR(32) | Configuration parameter name. |
| VALUE | VARCHAR(256) | The current value of the configuration parameter stored in memory. |
| VALUE_FLAGS | VARCHAR(10) | Provides specific information for the configuration parameter current value. Valid values are: <ul style="list-style-type: none"> • NONE - no additional information • AUTOMATIC - the configuration parameter has been set to automatic |

Table 40. Information returned by the DBMCFG administrative view (continued)

| Column name | Data type | Description |
|----------------------|--------------|---|
| DEFERRED_VALUE | VARCHAR(256) | The value of the configuration parameter on disk. For some database manager configuration parameters, the database manager must be stopped (db2stop) and restarted (db2start) 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"> • NONE - no additional information • AUTOMATIC - the configuration parameter has been set to automatic |
| DATATYPE | VARCHAR(128) | Configuration parameter data type. |

Related reference:

- “Configuration parameters summary” in *Performance Guide*
- “Administrative views versus table functions” on page 3
- “Authorization for administrative views” on page 6
- “DBCFCG administrative view – Retrieve database configuration parameter information” on page 182

REG_VARIABLES administrative view – Retrieve DB2 registry settings in use

The REG_VARIABLES administrative view returns the DB2 registry settings from all database partitions. The DB2 registry variable values returned when the REG_VARIABLES administrative view is queried can differ from those returned by the db2set command if a DB2 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.

The schema is SYSIBMADM.

Authorization:

SELECT or CONTROL privilege on the REG_VARIABLES administrative view and EXECUTE privilege on the REG_LIST_VARIABLES table function.

Example:

Request the DB2 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 41. Information returned by the REG_VARIABLES administrative view

| Column name | Data type | Description |
|----------------|---------------|---|
| DBPARTITIONNUM | SMALLINT | Logical partition number of each database partition on which the function operates. |
| REG_VAR_NAME | VARCHAR(256) | Name of the DB2 registry variable. |
| REG_VAR_VALUE | VARCHAR(2048) | Current setting of the DB2 registry variable. |
| IS_AGGREGATE | SMALLINT | Indicates whether or not the DB2 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. |

REG_VARIABLES

Table 41. Information returned by the REG_VARIABLES administrative view (continued)

| Column name | Data type | Description |
|----------------|--------------|--|
| AGGREGATE_NAME | VARCHAR(256) | 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 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 DB2 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 |

Related concepts:

- “DB2 registry and environment variables” in *Performance Guide*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “db2set - DB2 profile registry command” in *Command Reference*
- “Administrative views versus table functions” on page 3
- “Authorization for administrative views” on page 6

Environment administrative SQL routines and views

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:

SELECT or CONTROL privilege on the ENV_INST_INFO administrative view and EXECUTE privilege on the ENV_GET_INST_INFO table function.

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 v9.1.0.115        n051106                               0
...
```

Information returned:

Table 42. 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). |

ENV_INST_INFO

Table 42. Information returned by the ENV_INST_INFO administrative view (continued)

| Column name | Data type | Description |
|---------------|--------------|--|
| 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 v8.1.1.80. |
| 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. |

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Administrative views versus table functions” on page 3
- “Authorization for administrative views” on page 6
- “ENV_PROD_INFO administrative view – Retrieve information about installed DB2 products” on page 191
- “ENV_SYS_INFO administrative view – Retrieve information about the system” on page 193

ENV_PROD_INFO administrative view – Retrieve information about installed DB2 products

The ENV_PROD_INFO administrative view returns information about installed DB2 products.

The schema is SYSIBMADM.

Authorization:

SELECT or CONTROL privilege on the ENV_PROD_INFO administrative view and EXECUTE privilege on the ENV_GET_PROD_INFO table function.

Example:

Request the installed DB2 product information.

```
SELECT * FROM SYSIBMADM.ENV_PROD_INFO
```

The following is an example of output from this query.

```
INSTALLED_PROD    IS_LICENSED  PROD_RELEASE
-----...-----
ESE                1 9.1
WSE                1 9.1
PE                 1 9.1
CONPE              1 9.1
```

4 record(s) selected.

ENV_PROD_INFO administrative view metadata:

Table 43. ENV_PROD_INFO administrative view metadata

| Column name | Data type | Description |
|----------------|-------------|--|
| INSTALLED_PROD | VARCHAR(26) | Identifiers for one or more DB2 products that are installed on the system. The possible return values and the corresponding DB2 products that they represent are: <ul style="list-style-type: none"> • RTCL: DB2Run-Time Client • CLIENT: DB2 Client • * CONSV: DB2 Connect™ Server • CONPE: DB2 Connect Personal Edition • ESE: DB2 Enterprise Server Edition • EXP: DB2 Express Edition • PE: DB2 Personal Edition • WSE: DB2 Workgroup Server Edition * CONSV is returned when any DB2 Connect server edition is found. |
| IS_LICENSED | SMALLINT | Indicates whether or not the installed product is licensed. Possible return values are 0 (if the product is not licensed) and 1 (if it is licensed). |
| PROD_RELEASE | VARCHAR(26) | Product release number. |

Related reference:

ENV_PROD_INFO

- “Supported administrative SQL routines and views” on page 8
- “Administrative views versus table functions” on page 3
- “Authorization for administrative views” on page 6
- “ENV_INST_INFO administrative view – Retrieve information about the current instance” on page 189
- “ENV_SYS_INFO administrative view – Retrieve information about the system” on page 193

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:

SELECT or CONTROL privilege on the ENV_SYS_INFO administrative view and EXECUTE privilege on the ENV_GET_SYS_INFO table function.

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 44. 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) | Name of the system. |
| TOTAL_CPUS | INTEGER | Total number of physical CPUs on the system. |
| CONFIGURED_CPUS | INTEGER | Number of configured physical CPUs on the system. |
| TOTAL_MEMORY | INTEGER | Total amount of memory on the system (in megabytes). |

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Administrative views versus table functions” on page 3
- “Authorization for administrative views” on page 6

ENV_SYS_INFO

- “ENV_INST_INFO administrative view – Retrieve information about the current instance” on page 189
- “ENV_PROD_INFO administrative view – Retrieve information about installed DB2 products” on page 191

Health snapshot administrative SQL routines and views

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.

Syntax:

```
►►—HEALTH_CONT_HI—(—dbname—,—dbpartitionnum—)—————►►
```

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.

dbpartitionnum

An 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 database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_CONT_HI table function.

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).

HEALTH_CONT_HI

```

... 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 45. Information returned by the HEALTH_CONT_HI table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container Name monitor element |
| NODE_NUMBER | INTEGER | node_number - Node Number 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. |

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*

Related reference:

- “Health monitor SQL table functions” in *System Monitor Guide and Reference*
- “Supported administrative SQL routines and views” on page 8

HEALTH_CONT_HI_HIS

The HEALTH_CONT_HI_HIS table function returns health indicator history information for containers from a health snapshot of a database.

Syntax:

```
▶▶ HEALTH_CONT_HI_HIS ( (—dbname—, —dbpartitionnum—) ) ▶▶
```

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.

dbpartitionnum

An 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 database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_CONT_HI_HIS table function.

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 ...
```


HEALTH_CONT_HI_HIS

```

...      -      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 ...

```

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                -
... Normal                1                -

```

Information returned:

Table 46. Information returned by the HEALTH_CONT_HI_HIS table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container Name monitor element |
| NODE_NUMBER | INTEGER | node_number - Node Number monitor element |
| 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. |

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*

Related reference:

- “Health monitor SQL table functions” in *System Monitor Guide and Reference*
- “Supported administrative SQL routines and views” on page 8

HEALTH_CONT_INFO

The HEALTH_CONT_INFO table function returns container information from a health snapshot of a database.

Syntax:

```
▶▶ HEALTH_CONT_INFO (—dbname—, —dbpartitionnum—) ▶▶
```

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.

dbpartitionnum

An 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 database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_CONT_INFO table function.

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 |

HEALTH_CONT_INFO

... 1 Normal
... 1 Normal
... 1 Normal
... 1 Normal

Information returned:

Table 47. Information returned by the HEALTH_CONT_INFO table function

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container Name monitor element |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |
| NODE_NUMBER | INTEGER | node_number - Node Number monitor element |
| 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. |

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*

Related reference:

- “Health monitor SQL table functions” in *System Monitor Guide and Reference*
- “Supported administrative SQL routines and views” on page 8

HEALTH_DB_HI

The HEALTH_DB_HI table function returns health indicator information from a health snapshot of a database.

Syntax

```
▶▶—HEALTH_DB_HI—(—dbname—,—dbpartitionnum—)————▶▶
```

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.

dbpartitionnum

An 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 database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_DB_HI table function.

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 | ... |

HEALTH_DB_HI

```
... 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", userexit 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 48. Information returned by the HEALTH_DB_HI table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| DB_NAME | VARCHAR(128) | db_name - Database Name monitor element |
| HI_VALUE | SMALLINT | The value of the health indicator. |

Table 48. Information returned by the HEALTH_DB_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. |

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*

Related reference:

- “Health monitor SQL table functions” in *System Monitor Guide and Reference*
- “Supported administrative SQL routines and views” on page 8

HEALTH_DB_HI_HIS

The HEALTH_DB_HI_HIS table function returns health indicator history information from a health snapshot of a database.

Syntax:

```
▶▶ HEALTH_DB_HI_HIS (—dbname—, —dbpartitionnum—) ▶▶
```

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.

dbpartitionnum

An 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 database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_DB_HI_HIS table function.

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).

HEALTH_DB_HI_HIS

| 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 |
| $((0 / 5000) * 100)$ |
| $((0 - 0) / ((68 - 0) + 1)) * 100)$ |
| $((698410 / (698410 + 19701590)) * 100)$ |
| $((3920 / (50 * 4096)) * 100)$ |
| $((0 / 4) * 100)$ |
| $((4521984 / 6160384) * 100)$ |
| - |

```

... -
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...
...

```

Output from this query (continued).

```

... 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".
...
...

```

HEALTH_DB_HI_HIS

```
... The scope is "N\A". Automatic
... statistics collection
... (AUTO_RUNSTATS) is set to "OFF".
...
```

Information returned:

Table 49. Information returned by the HEALTH_DB_HI_HIS table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| DB_NAME | VARCHAR(128) | db_name - Database Name 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. |

Related concepts:

- "Health monitor" in *System Monitor Guide and Reference*

Related reference:

- "Health monitor SQL table functions" in *System Monitor Guide and Reference*
- "Supported administrative SQL routines and views" on page 8

HEALTH_DB_HIC

The HEALTH_DB_HIC function returns collection health indicator information from a health snapshot of a database.

Syntax:

```
▶▶ HEALTH_DB_HIC (—dbname—, —dbpartitionnum—) ▶▶
```

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_DB_HIC table function.

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 50. Information returned by the HEALTH_DB_HIC table function*

| Column name | Data type | Description or corresponding monitor element |
|---------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| DB_NAME | VARCHAR(128) | db_name - Database Name monitor element |
| HI_OBJ_NAME | VARCHAR(256) | A name that uniquely identifies an object in the collection. |
| HI_OBJ_DETAIL | VARCHAR(4096) | Text that describes why the object was added to the collection. |
| HI_OBJ_STATE | BIGINT | 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. |

Related concepts:

- "Health monitor" in *System Monitor Guide and Reference*

Related reference:

- "Supported administrative SQL routines and views" on page 8
- "Health monitor SQL table functions" in *System Monitor Guide and Reference*

HEALTH_DB_HIC_HIS

The HEALTH_DB_HIC_HIS table function returns collection health indicator history information from a health snapshot of a database.

Syntax:

```
▶▶—HEALTH_DB_HIC_HIS—(—dbname—,—dbpartitionnum—)————▶▶
```

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_DB_HIC_HIS table function.

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 | ... |

HEALTH_DB_HIC_HIS

```

... 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 51. Information returned by the HEALTH_DB_HIC_HIS table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| HI_HIS_ENTRY_NUM | INTEGER | A number that uniquely identifies the history entry. |
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |
| DB_NAME | VARCHAR(128) | db_name - Database Name monitor element |
| HI_OBJ_NAME | VARCHAR(256) | A name that uniquely identifies an object in the collection. |

Table 51. Information returned by the HEALTH_DB_HIC_HIS table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|-------------|---|
| HI_OBJ_STATE | BIGINT | 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. |

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Health monitor SQL table functions” in *System Monitor Guide and Reference*

HEALTH_DB_INFO

The HEALTH_DB_INFO table function returns information from a health snapshot of a database.

Syntax:

```
▶▶ HEALTH_DB_INFO (—dbname—, —dbpartitionnum—) ▶▶
```

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.

dbpartitionnum

An 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 database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_DB_INFO table function.

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 52. Information returned by the HEALTH_DB_INFO 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 monitor element |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input Database Alias monitor element |
| DB_PATH | VARCHAR(1024) | db_path - Database Path monitor element |
| DB_LOCATION | INTEGER | db_location - Database Location monitor element |
| SERVER_PLATFORM | INTEGER | server_platform - Server Operating System monitor element |
| 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. |

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*

Related reference:

- “Health monitor SQL table functions” in *System Monitor Guide and Reference*
- “Supported administrative SQL routines and views” on page 8

HEALTH_DBM_HI

The HEALTH_DBM_HI table function returns health indicator information from a health snapshot of the DB2 database manager.

Syntax:

```
▶▶ HEALTH_DBM_HI(—dbpartitionnum—)▶▶
```

The schema is SYSPROC.

Table function parameter:

dbpartitionnum

An 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 database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_DBM_HI table function.

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 53. Information returned by the HEALTH_DBM_HI table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| HI_ID | BIGINT | A number that uniquely identifies the health indicator in the snapshot data stream. |

Table 53. Information returned by the HEALTH_DBM_HI table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SERVER_INSTANCE_NAME | VARCHAR(128) | server_instance_name - Server Instance Name 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. |

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*

Related reference:

- “Health monitor SQL table functions” in *System Monitor Guide and Reference*
- “Supported administrative SQL routines and views” on page 8

HEALTH_DBM_HI_HIS

The HEALTH_DBM_HI_HIS table function returns health indicator history information from a health snapshot of the DB2 database manager.

Syntax:

```
▶▶ HEALTH_DBM_HI_HIS (—dbpartitionnum—) ▶▶
```

The schema is SYSPROC.

Table function parameter:

dbpartitionnum

An 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 database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_DBM_HI_HIS table function.

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                          -
... 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 54. Information returned by the HEALTH_DBM_HI_HIS table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| 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 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_HI_HIS

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*

Related reference:

- “Health monitor SQL table functions” in *System Monitor Guide and Reference*
- “Supported administrative SQL routines and views” on page 8

HEALTH_DBM_INFO

The HEALTH_DBM_INFO function returns information from a health snapshot of the DB2 database manager.

Syntax:

```
▶▶—HEALTH_DBM_INFO—(—dbpartitionnum—)————▶▶
```

The schema is SYSPROC.

Table function parameter:

dbpartitionnum

An 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 database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_DBM_INFO table function.

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 55. Information returned by the HEALTH_DBM_INFO table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| SERVER_INSTANCE_NAME | VARCHAR(128) | server_instance_name - Server Instance Name monitor element |
| ROLLED_UP_ALERT_STATE | BIGINT | The most severe alert state captured by this snapshot. |

HEALTH_DBM_INFO

Table 55. Information returned by the HEALTH_DBM_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. |
| DB2START_TIME | TIMESTAMP | db2start_time - Start Database Manager Timestamp monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |
| NUM_NODES_IN_DB2_INSTANCE | INTEGER | num_nodes_in_db2_instance - Number of Nodes in Partition monitor element |

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*

Related reference:

- “Health monitor SQL table functions” in *System Monitor Guide and Reference*
- “Supported administrative SQL routines and views” on page 8

HEALTH_GET_ALERT_ACTION_CFG table function –Retrieve health alert action configuration settings

The HEALTH_GET_ALERT_ACTION_CFG table function 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).

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:

EXECUTE privilege on the HEALTH_GET_ALERT_ACTION_CFG table function.

Examples:

Example 1: Retrieve object level alert action configuration settings for database SAMPLE for health indicator ID 1004.

HEALTH_GET_ALERT_ACTION_CFG

```
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 56. 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. |
| ID | BIGINT | Health indicator ID. |
| 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(512) | 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) | Host system name. |
| SCRIPT_TYPE | CHAR(1) | Script type: If ACTIONTYPE is 'S', 'O' for operating system command script or 'D' for DB2 command script; If ACTIONTYPE is 'T', Null. |
| WORKING_DIR | VARCHAR(5000) | The working directory for the script if ACTIONTYPE is 'S' or Null if ACTIONTYPE is 'T'. |
| TERMINATION_CHAR | VARCHAR(4) | The statement termination character if it is a DB2 command script action, otherwise Null. |
| PARAMETERS | VARCHAR(200) | The command line parameters if it is an operating system command script action. |

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “HEALTH_GET_ALERT_CFG table function – Retrieve health alert configuration settings” on page 226
- “HEALTH_GET_IND_DEFINITION table function – Retrieve health indicator definitions” on page 230

HEALTH_GET_ALERT_CFG table function – Retrieve health alert configuration settings

The HEALTH_GET_ALERT_CFG table function 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).

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:

EXECUTE privilege on the HEALTH_GET_ALERT_CFG table function.

Examples:

Example 1: Retrieve the object level alert configuration settings for database SAMPLE.

HEALTH_GET_ALERT_CFG

```
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 | | ... |
| 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
```

HEALTH_GET_ALERT_CFG

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 57. Information returned by the HEALTH_GET_ALERT_CFG table function

| Column name | Data type | Description |
|-------------------|--------------|--|
| OBJECTTYPE | VARCHAR(3) | Object type. |
| CFG_LEVEL | VARCHAR(1) | Configuration level. |
| DBNAME | VARCHAR(128) | Database name. |
| OBJECTNAME | VARCHAR(512) | Object name. |
| ID | BIGINT | Health indicator ID. |
| 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. |

Table 57. Information returned by the HEALTH_GET_ALERT_CFG table function (continued)

| Column name | Data type | Description |
|----------------|-----------|---|
| 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. |

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*
- “Health indicator configuration” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “HEALTH_GET_ALERT_ACTION_CFG table function – Retrieve health alert action configuration settings” on page 223
- “HEALTH_GET_IND_DEFINITION table function – Retrieve health indicator definitions” on page 230

HEALTH_GET_IND_DEFINITION table function – Retrieve health indicator definitions

The HEALTH_GET_IND_DEFINITION table function returns the health indicator definitions.

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:

EXECUTE privilege on the HEALTH_GET_IND_DEFINITION table function.

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 58. Information returned by the HEALTH_GET_IND_DEFINITION table function*

| Column name | Data type | Description |
|-------------------|----------------|---|
| ID | BIGINT | Health indicator ID. |
| 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. |

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “HEALTH_GET_ALERT_ACTION_CFG table function – Retrieve health alert action configuration settings” on page 223
- “HEALTH_GET_ALERT_CFG table function – Retrieve health alert configuration settings” on page 226
- “Health indicators” in *System Monitor Guide and Reference*

HEALTH_GET_IND_DEFINITION

- “Health indicators summary” in *System Monitor Guide and Reference*

HEALTH_HI_REC

```

▶▶—HEALTH_HI_REC—(—schema-version—,—indicator-id—,—dbname—,——————▶
▶—object-type—,—object-name—,—dbpartitionnum—,—client-locale—,—————▶
▶—recommendation-doc—)—————▶▶

```

The schema is SYSPROC.

The HEALTH_HI_REC procedure retrieves a set of recommendations that address a health indicator in alert state on a particular DB2 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. 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).

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

HEALTH_HI_REC

to 999, -1 (which specifies the currently connected database partition), and -2 (which specifies all database partitions).

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 DB2 Health Recommendation schema definition (see the XML schema DB2RecommendationSchema.xsd, located in the misc subdirectory of the sqllib 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.

Related concepts:

- "Health monitor" in *System Monitor Guide and Reference*

Related reference:

- "HEALTH_CONT_HI " on page 195
- "HEALTH_CONT_HI_HIS " on page 197
- "HEALTH_DB_HI " on page 201
- "HEALTH_DB_HI_HIS " on page 205
- "HEALTH_DBM_HI " on page 216
- "HEALTH_DBM_HI_HIS " on page 218
- "HEALTH_TBS_HI " on page 235
- "HEALTH_TBS_HI_HIS " on page 238
- "Health monitor SQL table functions" in *System Monitor Guide and Reference*
- "Supported administrative SQL routines and views" on page 8

HEALTH_TBS_HI

The HEALTH_TBS_HI table function returns health indicator information for table spaces from a health snapshot of table spaces in a database.

Syntax:

```
▶▶ HEALTH_TBS_HI(—dbname—,—dbpartitionnum—)▶▶
```

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.

dbpartitionnum

An 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 database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_TBS_HI table function.

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 | ... |

HEALTH_TBS_HI

```

... 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" respectively. 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" respectively. 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" respectively. 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 59. Information returned by the HEALTH_TBS_HI table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table Space Name 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. |

Related concepts:

- "Health monitor" in *System Monitor Guide and Reference*

Related reference:

- "Health monitor SQL table functions" in *System Monitor Guide and Reference*
- "Supported administrative SQL routines and views" on page 8

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.

Syntax

```
▶▶—HEALTH_TBS_HI_HIS—(—dbname—,—dbpartitionnum—)—————▶▶
```

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.

dbpartitionnum

An 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 database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_TBS_HI_HIS table function.

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" respectively. 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" respectively. 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) |

HEALTH_TBS_HI_HIS

```

... 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" respectively. 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
"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" respectively. 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 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 -
... 0 -
... 0 -
... 0 -
... ((1504 / 1504) * 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"
respectively. The table space is defined
with automatic storage set to "YES" and
automatic resize enabled set to "YES".
... ((1504 / 1504) * 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" respectively. 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".

... 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 60. Information returned by the HEALTH_TBS_HI_HIS table function*

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |
| 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. |

Related concepts:

- "Health monitor" in *System Monitor Guide and Reference*

Related reference:

- "Health monitor SQL table functions" in *System Monitor Guide and Reference*
- "Supported administrative SQL routines and views" on page 8

HEALTH_TBS_INFO

The HEALTH_TBS_INFO table function returns table space information from a health snapshot of a database.

Syntax:

```
▶▶ HEALTH_TBS_INFO ( (—dbname—, —dbpartitionnum—) ) ▶▶
```

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.

dbpartitionnum

An 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 database partitions. If the null value is specified, -1 is set implicitly.

Authorization:

EXECUTE privilege on the HEALTH_TBS_INFO table function.

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 61. Information returned by the HEALTH_TBS_INFO table function*

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |
| 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. |

Related concepts:

- “Health monitor” in *System Monitor Guide and Reference*

Related reference:

- “Health monitor SQL table functions” in *System Monitor Guide and Reference*
- “Supported administrative SQL routines and views” on page 8

MQSeries administrative SQL routines and views

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:

```

MQPUBLISH ( ( publisher-service , service-policy ) msg-data
            ( , topic )
            ( , correl-id (1) ) )

```

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.

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')
```

Related concepts:

- "WebSphere MQ and DB2 User Defined Functions" in *Application Development Guide for Federated Systems*

Related reference:

- "MQREAD " on page 247
- "MQREADALL " on page 249
- "MQREADALLCLOB " on page 252
- "MQREADCLOB " on page 255
- "MQRECEIVE " on page 257
- "MQRECEIVEALLCLOB " on page 262
- "MQRECEIVECLOB " on page 265
- "MQSEND " on page 267
- "MQSUBSCRIBE " on page 269
- "MQUNSUBSCRIBE " on page 271
- "MQRECEIVEALL " on page 259

MQPUBLISH

- “Supported administrative SQL routines and views” on page 8

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.

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')
```

Related concepts:

MQREAD

- “WebSphere MQ and DB2 User Defined Functions” in *Application Development Guide for Federated Systems*

Related reference:

- “MQREADALLCLOB ” on page 252
- “MQREADCLOB ” on page 255
- “MQRECEIVE ” on page 257
- “MQPUBLISH ” on page 244
- “MQREADALL ” on page 249
- “MQSUBSCRIBE ” on page 269
- “MQUNSUBSCRIBE ” on page 271
- “MQRECEIVEALL ” on page 259
- “MQRECEIVEALLCLOB ” on page 262
- “MQRECEIVECLOB ” on page 265
- “MQSEND ” on page 267
- “Supported administrative SQL routines and views” on page 8

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:

EXECUTE privilege on the MQREADALL table function.

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
```

MQREADALL

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 62. Information returned by the MQREADALL 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. |

Related concepts:

- “WebSphere MQ and DB2 User Defined Functions” in *Application Development Guide for Federated Systems*

Related reference:

- “MQPUBLISH ” on page 244
- “MQREAD ” on page 247

- “MQREADALLCLOB ” on page 252
- “MQREADCLOB ” on page 255
- “MQRECEIVE ” on page 257
- “MQRECEIVEALL ” on page 259
- “MQRECEIVEALLCLOB ” on page 262
- “MQRECEIVECLOB ” on page 265
- “MQSEND ” on page 267
- “MQSUBSCRIBE ” on page 269
- “MQUNSUBSCRIBE ” on page 271
- “Supported administrative SQL routines and views” on page 8

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:



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:

EXECUTE privilege on the MQREADALLCLOB table function.

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 63. 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. |

Related concepts:

- “WebSphere MQ and DB2 User Defined Functions” in *Application Development Guide for Federated Systems*

Related reference:

- “MQPUBLISH ” on page 244
- “MQREAD ” on page 247

MQREADALLCLOB

- “MQREADALL ” on page 249
- “MQREADCLOB ” on page 255
- “MQRECEIVE ” on page 257
- “MQRECEIVEALL ” on page 259
- “MQRECEIVEALLCLOB ” on page 262
- “MQRECEIVECLOB ” on page 265
- “MQSEND ” on page 267
- “MQSUBSCRIBE ” on page 269
- “MQUNSUBSCRIBE ” on page 271
- “Supported administrative SQL routines and views” on page 8

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.

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')
```

Related concepts:

MQREADCLOB

- “WebSphere MQ and DB2 User Defined Functions” in *Application Development Guide for Federated Systems*

Related reference:

- “MQPUBLISH ” on page 244
- “MQREAD ” on page 247
- “MQREADALL ” on page 249
- “MQREADALLCLOB ” on page 252
- “MQRECEIVE ” on page 257
- “MQRECEIVEALL ” on page 259
- “MQRECEIVEALLCLOB ” on page 262
- “MQRECEIVECLOB ” on page 265
- “MQSEND ” on page 267
- “MQSUBSCRIBE ” on page 269
- “MQUNSUBSCRIBE ” on page 271
- “Supported administrative SQL routines and views” on page 8

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:

```

MQRECEIVE (
  receive-service
  [, service-policy
  [, correl-id
)

```

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.

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()
```

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')
```

Related concepts:

- "WebSphere MQ and DB2 User Defined Functions" in *Application Development Guide for Federated Systems*

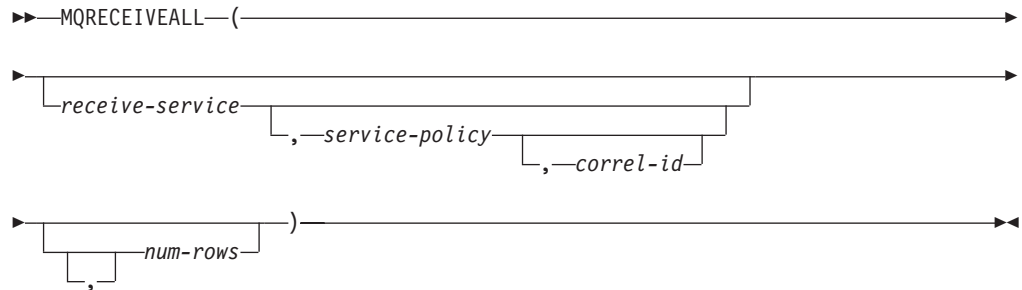
Related reference:

- "MQREADCLOB " on page 255
- "MQRECEIVEALL " on page 259
- "MQRECEIVEALLCLOB " on page 262
- "MQRECEIVECLOB " on page 265
- "MQSEND " on page 267
- "MQPUBLISH " on page 244
- "MQREAD " on page 247
- "MQREADALL " on page 249
- "MQREADALLCLOB " on page 252
- "MQSUBSCRIBE " on page 269
- "MQUNSUBSCRIBE " on page 271
- "Supported administrative SQL routines and views" on page 8

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.

MQRECEIVEALL

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:

EXECUTE privilege on the MQRECEIVEALL table function.

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 64. 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. |

Table 64. Information returned by the MQRECEIVEALL table function (continued)

| Column name | Data type | Description |
|-------------|------------|---|
| 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. |

Related concepts:

- “WebSphere MQ and DB2 User Defined Functions” in *Application Development Guide for Federated Systems*

Related reference:

- “MQREADALL ” on page 249
- “MQREADALLCLOB ” on page 252
- “MQPUBLISH ” on page 244
- “MQREAD ” on page 247
- “MQREADCLOB ” on page 255
- “MQRECEIVE ” on page 257
- “MQRECEIVEALLCLOB ” on page 262
- “MQRECEIVECLOB ” on page 265
- “MQSEND ” on page 267
- “MQSUBSCRIBE ” on page 269
- “MQUNSUBSCRIBE ” on page 271
- “Supported administrative SQL routines and views” on page 8

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
)
  num-rows

```

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:

EXECUTE privilege on the MQRECEIVEALLCLOB table function.

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 65. 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. |

MQRECEIVEALLCLOB

Table 65. Information returned by the MQRECEIVEALLCLOB table function (continued)

| Column name | Data type | Description |
|-------------|------------|---|
| MSGFORMAT | VARCHAR(8) | Contains the format of the message, as defined by MQSeries. Typical strings have an MQSTR format. |

Related concepts:

- “WebSphere MQ and DB2 User Defined Functions” in *Application Development Guide for Federated Systems*

Related reference:

- “MQRECEIVECLOB ” on page 265
- “MQSEND ” on page 267
- “MQUNSUBSCRIBE ” on page 271
- “MQPUBLISH ” on page 244
- “MQREAD ” on page 247
- “MQREADALL ” on page 249
- “MQREADALLCLOB ” on page 252
- “MQREADCLOB ” on page 255
- “MQRECEIVE ” on page 257
- “MQRECEIVEALL ” on page 259
- “Supported administrative SQL routines and views” on page 8

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:

```

MQRECEIVECLOB
(
  receive-service
  , service-policy
  , correl-id
)

```

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.

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()
```

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')
```

Related concepts:

- "WebSphere MQ and DB2 User Defined Functions" in *Application Development Guide for Federated Systems*

Related reference:

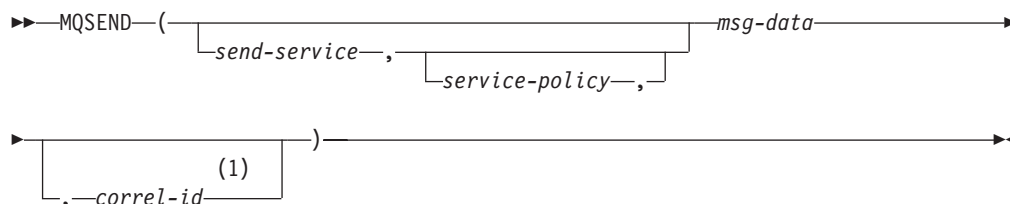
- "MQRECEIVEALL " on page 259
- "MQRECEIVEALLCLOB " on page 262
- "MQPUBLISH " on page 244
- "MQREAD " on page 247
- "MQREADALL " on page 249
- "MQREADALLCLOB " on page 252
- "MQREADCLOB " on page 255
- "MQRECEIVE " on page 257
- "MQSEND " on page 267
- "MQSUBSCRIBE " on page 269
- "MQUNSUBSCRIBE " on page 271
- "Supported administrative SQL routines and views" on page 8

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.

MQSEND

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')
```

Related concepts:

- "WebSphere MQ and DB2 User Defined Functions" in *Application Development Guide for Federated Systems*

Related reference:

- "MQPUBLISH " on page 244
- "MQREAD " on page 247
- "MQREADALL " on page 249
- "MQREADALLCLOB " on page 252
- "MQREADCLOB " on page 255
- "MQRECEIVE " on page 257
- "MQRECEIVEALL " on page 259
- "MQRECEIVEALLCLOB " on page 262
- "MQRECEIVECLOB " on page 265
- "MQSUBSCRIBE " on page 269
- "MQUNSUBSCRIBE " on page 271
- "Supported administrative SQL routines and views" on page 8

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.

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.

MQSUBSCRIBE

```
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')
```

Related concepts:

- "WebSphere MQ and DB2 User Defined Functions" in *Application Development Guide for Federated Systems*

Related reference:

- "MQPUBLISH " on page 244
- "MQREAD " on page 247
- "MQREADALL " on page 249
- "MQREADALLCLOB " on page 252
- "MQREADCLOB " on page 255
- "MQRECEIVE " on page 257
- "MQRECEIVEALL " on page 259
- "MQRECEIVEALLCLOB " on page 262
- "MQRECEIVECLOB " on page 265
- "MQSEND " on page 267
- "MQUNSUBSCRIBE " on page 271
- "Supported administrative SQL routines and views" on page 8

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 canceled. 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:

```

MQUNSUBSCRIBE
(
  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

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".

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')
```

MQUNSUBSCRIBE

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')
```

Related concepts:

- "WebSphere MQ and DB2 User Defined Functions" in *Application Development Guide for Federated Systems*

Related reference:

- "MQPUBLISH " on page 244
- "MQREAD " on page 247
- "MQREADALL " on page 249
- "MQREADALLCLOB " on page 252
- "MQREADCLOB " on page 255
- "MQRECEIVE " on page 257
- "MQRECEIVEALL " on page 259
- "MQRECEIVEALLCLOB " on page 262
- "MQRECEIVECLOB " on page 265
- "MQSEND " on page 267
- "MQSUBSCRIBE " on page 269
- "Supported administrative SQL routines and views" on page 8

Security administrative SQL routines and views

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, an empty result table is returned.

Authorization:

EXECUTE privilege on the AUTH_LIST_GROUPS_FOR_AUTHID table function.

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.

AUTH_LIST_GROUPS_FOR_AUTHID

- 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.

Information returned:

Table 66. 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. |

Related concepts:

- “Authorization” in *Administration Guide: Planning*

Related tasks:

- “Authentication with groups and domain security (Windows)” in *Administration Guide: Implementation*
- “Retrieving authorization names with granted privileges” in *Administration Guide: Implementation*

Related reference:

- “Supported administrative SQL routines and views” on page 8

AUTHORIZATIONIDS administrative view – Retrieve authorization IDs and types

The AUTHORIZATIONIDS administrative view returns a list of authorization IDs that have been granted privileges or authorities, along with their types, for all authorization IDs defined in the system catalogs from the currently connected database. If privileges or authorities have been granted to groups, only the group names are returned.

The schema is SYSIBMADM.

Authorization:

SELECT or CONTROL privilege on the AUTHORIZATIONIDS administrative view.

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
```

2 record(s) selected.

Information returned:

Table 67. 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 • G: group |

Related concepts:

- “Authorization, privileges, and object ownership” in *Administration Guide: Implementation*

Related tasks:

- “Retrieving authorization names with granted privileges” in *Administration Guide: Implementation*
- “Securing the system catalog view” in *Administration Guide: Implementation*

Related reference:

- “Supported administrative SQL routines and views” on page 8

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:

SELECT or CONTROL privilege on the OBJECTOWNERS administrative view.

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 OJECTSCHEMA='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 | SQLQ0F00 | NULLID | PACKAGE |
| ... | | | | |
| THERESAX | U | HI_OBJ_UNIQ | SYSTOOLS | TABLE CONSTRAINT |

257 record(s) selected.

Information returned:

Table 68. 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: • U: user |
| OBJECTNAME | VARCHAR(128) | Database object name. |
| OBJECTSCHEMA | VARCHAR(128) | Database object schema. |
| OBJECTTYPE | VARCHAR(24) | Database object type. |

Related concepts:

- “Authorization, privileges, and object ownership” in *Administration Guide: Implementation*

Related tasks:

- “Retrieving authorization names with granted privileges” in *Administration Guide: Implementation*
- “Securing the system catalog view” in *Administration Guide: Implementation*

Related reference:

- “Supported administrative SQL routines and views” on page 8

PRIVILEGES

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:

SELECT or CONTROL privilege on the PRIVILEGES administrative view.

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 |
| ... | | | | |
| 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_SVR_TYPES | SYSPROC | PROCEDURE |
| PUBLIC | EXECUTE | LIST_SVR_VERSIONS | SYSPROC | PROCEDURE |
| PUBLIC | EXECUTE | LIST_WRAP_OPTIONS | SYSPROC | PROCEDURE |
| PUBLIC | EXECUTE | LIST_SVR_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 69. 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• G: group |

Table 69. Information returned by the PRIVILEGES administrative view (continued)

| Column name | Data type | Description |
|--------------|--------------|---|
| 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) | Database object name. |
| OBJECTSCHEMA | VARCHAR(128) | Database object schema. |
| OBJECTTYPE | VARCHAR(24) | Database object type. |

Related concepts:

- “Authorization, privileges, and object ownership” in *Administration Guide: Implementation*
- “Controlling access to database objects” in *Administration Guide: Implementation*

Related tasks:

- “Retrieving all privileges granted to users” in *Administration Guide: Implementation*
- “Retrieving authorization names with granted privileges” in *Administration Guide: Implementation*
- “Retrieving names authorized to access a table” in *Administration Guide: Implementation*

Related reference:

- “Supported administrative SQL routines and views” on page 8

Snapshot administrative SQL routines and views

APPLICATIONS administrative view – Retrieve connected database application information

The APPLICATIONS administrative view returns information on 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.

The schema is SYSIBMADM.

Authorization:

- SELECT or CONTROL privilege on the APPLICATIONS and SNAPAPPL_INFO administrative views.
- SYSMON, SYSCTRL, SYSMAINT, or SYSADM authority which is required to access snapshot monitor data.

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, AUTH_ID,
       APPL_STATUS FROM SYSIBMADM.APPLICATIONS WHERE DB_NAME = 'SAMPLE'
```

The following is an example of output for this query.

| AGENT_ID | APPL_NAME | AUTH_ID | 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 70. 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 monitor element |
| DB_NAME | VARCHAR(128) | db_name - Database Name monitor element |
| AGENT_ID | BIGINT | agent_id - Application Handle (agent ID) monitor element |
| APPL_NAME | VARCHAR(256) | appl_name - Application Name monitor element |
| AUTHID | VARCHAR(128) | auth_id - Authorization ID monitor element |
| APPL_ID | VARCHAR(128) | appl_id - Application ID monitor element |

APPLICATIONS

Table 70. 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 monitor element. 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 monitor element |
| SEQUENCE_NO | VARCHAR(4) | sequence_no - Sequence Number monitor element |
| CLIENT_PRDID | VARCHAR(128) | client_prdid - Client Product/Version ID monitor element |
| CLIENT_PID | BIGINT | client_pid - Client Process ID monitor element |

Table 70. 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 monitor element. 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 |

APPLICATIONS

Table 70. 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 monitor element. This interface returns a text identifier based on the defines in sqlmon.h, <ul style="list-style-type: none"> • APPC • APPN • CPIC • IPXSPX • LOCAL • NETBIOS • NPIPE • TCPIP (for DB2 UDB) • TCPIP4 • TCPIP6 |
| CLIENT_NNAME | VARCHAR(128) | client_nname - Configuration NNAME of Client monitor element |
| COORD_NODE_NUM | SMALLINT | coord_node - Coordinating Node monitor element |
| COORD_AGENT_PID | BIGINT | coord_agent_pid - Coordinator Agent monitor element |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of Associated Agents monitor element |
| TPMON_CLIENT_USERID | VARCHAR(256) | tpmon_client_userid - TP Monitor Client User ID monitor element |
| TPMON_CLIENT_WKSTN | VARCHAR(256) | tpmon_client_wkstn - TP Monitor Client Workstation Name monitor element |
| TPMON_CLIENT_APP | VARCHAR(256) | tpmon_client_app - TP Monitor Client Application Name monitor element |
| TPMON_ACC_STR | VARCHAR(200) | tpmon_acc_str - TP Monitor Client Accounting String monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “LIST APPLICATIONS command” in *Command Reference*
- “SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information” on page 334
- “Authorization for administrative views” on page 6

- “Database system monitor elements” in *System Monitor Guide and Reference*
- “FORCE APPLICATION command using the ADMIN_CMD procedure” on page 76

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.

The schema is SYSIBMADM.

Authorization:

- SELECT or CONTROL privilege on the APPL_PERFORMANCE, SNAPAPPL_INFO and SNAPAPPL administrative views.
- SYSMON, SYSCTRL, SYSMMAINT, or SYSADM authority are also required to access snapshot monitor data.

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 71. 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 monitor element |
| APPL_NAME | VARCHAR(256) | appl_name - Application Name monitor element |
| AGENT_ID | BIGINT | agent_id - Application Handle (agent ID) monitor element |

Table 71. Information returned by the APPL_PERFORMANCE administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|--------------|--|
| PERCENT_ROWS_SELECTED | DECIMAL(5,2) | The percent of rows read from disk that were actually returned to the application. |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information” on page 334
- “SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information” on page 324
- “Authorization for administrative views” on page 6
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

The schema is SYSIBMADM.

Authorization:

- SELECT or CONTROL privilege on the BP_HITRATIO and SNAPBP administrative views.
- SYSMON, SYSCTRL, SYSMOINT, or SYSADM authority is also required to access snapshot monitor data.

Example:

Retrieve a report for all bufferpools in the connected database.

```
SELECT SUBSTR(DB_NAME,1,8) AS DB_NAME, SUBSTR(BP_NAME,1,140) 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 logical 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 72. 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 monitor element |

Table 72. Information returned by the BP_HITRATIO administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|--|
| BP_NAME | VARCHAR(128) | bp_name - Buffer Pool Name monitor element |
| 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). |
| DATA_LOGICAL_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| DATA_PHYSICAL_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |
| DATA_HIT_RATIO_PERCENT | DECIMAL(5,2) | Data hit ratio. |
| INDEX_LOGICAL_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| INDEX_PHYSICAL_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |
| INDEX_HIT_RATIO_PERCENT | DECIMAL(5,2) | Index hit ratio. |
| XDA_LOGICAL_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads monitor element |
| XDA_PHYSICAL_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads monitor element |
| XDA_HIT_RATIO_PERCENT | DECIMAL(5,2) | Auxiliary storage objects hit ratio. |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data for the row was retrieved. |

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Database system monitor elements” in *System Monitor Guide and Reference*
- “Authorization for administrative views” on page 6
- “SNAPBP administrative view and SNAP_GET_BP table function – Retrieve bufferpool logical group snapshot information” on page 341

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.

The schema is SYSIBMADM.

Authorization:

- SELECT or CONTROL privilege on the BP_READ_IO and SNAPBP administrative views.
- SYSMON, SYSCTRL, SYSMOINT, or SYSADM authority are also required to access snapshot monitor data.

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 73. 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 monitor element |
| TOTAL_PHYSICAL_READS | BIGINT | Total physical reads. |
| AVERAGE_READ_TIME_MS | BIGINT | Average read time in milliseconds. |
| TOTAL_ASYNC_READS | BIGINT | Total asynchronous reads. |

Table 73. Information returned by the BP_READ_IO administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|--------------|--|
| 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. |
| 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 | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAPBP administrative view and SNAP_GET_BP table function – Retrieve bufferpool logical group snapshot information” on page 341
- “Authorization for administrative views” on page 6
- “Database system monitor elements” in *System Monitor Guide and Reference*

BP_WRITE_IO administrative view – Retrieve bufferpool write performance information

The BP_WRITE_IO administrative view returns bufferpool write performance information per bufferpool.

The schema is SYSIBMADM.

Authorization:

- SELECT or CONTROL privilege on the BP_WRITE_IO and SNAPBP administrative views.
- SYSMON, SYSCTRL, SYSMOINT, or SYSADM authority is also required to access snapshot monitor data.

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 74. 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 monitor element |
| TOTAL_WRITES | BIGINT | Total writes. |
| AVERAGE_WRITE_TIME_MS | BIGINT | Average write time in milliseconds. |
| TOTAL_ASYNC_WRITES | BIGINT | Total asynchronous writes. |

Table 74. Information returned by the BP_WRITE_IO administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------------|-----------|---|
| PERCENT_WRITES_ASYNC | BIGINT | Percent of writes that are asynchronous. |
| 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 | The database partition from which the data for the row was retrieved. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAPBP administrative view and SNAP_GET_BP table function – Retrieve bufferpool logical group snapshot information” on page 341
- “Authorization for administrative views” on page 6
- “Database system monitor elements” in *System Monitor Guide and Reference*

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. The view is an SQL interface for the **LIST TABLESPACE CONTAINERS** CLP command. Its information is based on the SNAPCONTAINER administrative view.

The schema is SYSIBMADM.

Authorization:

- SELECT or CONTROL privilege on CONTAINER_UTILIZATION and SNAPCONTAINER administrative views.
- SYSMON, SYSCTRL, SYSMAINT, or SYSADM authority (required to access snapshot monitor data).

Example:

Retrieve a list of all table spaces containers in the connected single partition database, including information on 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 75. 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 monitor element |
| TBSP_ID | BIGINT | tablespace_id - Table Space Identification monitor element |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container Name monitor element |
| CONTAINER_ID | BIGINT | container_id - Container Identification monitor element |
| CONTAINER_TYPE | VARCHAR(16) | container_type - Container Type monitor element. 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 monitor element |
| USABLE_PAGES | BIGINT | container_usable_pages - Usable Pages in Container monitor element |
| ACCESSIBLE | SMALLINT | container_accessible - Accessibility of Container monitor element |
| STRIPE_SET | BIGINT | container_stripe_set - Stripe Set monitor element |
| FS_ID | VARCHAR(22) | fs_id - Unique File System Identification Number monitor element |
| FS_TOTAL_SIZE_KB | BIGINT | fs_total_size - Total Size of a File System monitor element. This interface returns the value in KB. |
| FS_USED_SIZE_KB | BIGINT | fs_used_size - Amount of Space Used on a File System monitor element. This interface returns the value in KB. |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Authorization for administrative views” on page 6
- “Database system monitor elements” in *System Monitor Guide and Reference*

CONTAINER_UTILIZATION

- “LIST TABLESPACE CONTAINERS command” in *Command Reference*

LOCKS_HELD administrative view – Retrieve information on locks held

The LOCKS_HELD administrative view returns information on current locks held.

The schema is SYSIBMADM.

Authorization:

- SELECT or CONTROL privilege on the LOCKS_HELD, SNAPLOCK and SNAPAPPL_INFO administrative views.
- SYSMON, SYSCTRL, SYSMOINT, or SYSADM authority is also required to access snapshot monitor data.

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 |

LOCKS_HELD

Information returned:

Table 76. 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 monitor element |
| AGENT_ID | BIGINT | agent_id - Application Handle (agent ID) monitor element |
| APPL_NAME | VARCHAR(256) | appl_name - Application Name monitor element |
| AUTHID | VARCHAR(128) | auth_id - Authorization ID monitor element |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |
| TABSCHEMA | VARCHAR(128) | table_schema - Table Schema Name monitor element |
| TABNAME | VARCHAR(128) | table_name - Table Name monitor element |
| TAB_FILE_ID | BIGINT | table_file_id - Table File ID monitor element |

Table 76. 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 monitor element. 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_NAME | VARCHAR(32) | lock_name - Lock Name monitor element |

LOCKS_HELD

Table 76. Information returned by the LOCKS_HELD administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------|-------------|--|
| LOCK_MODE | VARCHAR(10) | lock_mode - Lock Mode monitor element. 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• NX• S• SIX• U• W• X• Z |
| LOCK_STATUS | VARCHAR(10) | lock_status - Lock Status monitor element. 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 monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information” on page 334
- “SNAPLOCK administrative view and SNAP_GET_LOCK table function – Retrieve lock logical data group snapshot information” on page 403
- “Authorization for administrative views” on page 6
- “Database system monitor elements” in *System Monitor Guide and Reference*

LOCKWAITS administrative view – Retrieve current lockwaits information

The LOCKWAITS administrative view returns information about DB2 agents working on behalf of applications that are waiting to obtain locks.

The schema is SYSIBMADM.

Authorization:

- SELECT or CONTROL privilege on the LOCKWAITS, SNAPAPPL_INFO and SNAPLOCKWAIT administrative views.
- SYSMON, SYSCTRL, SYSMOINT, or SYSADM authority is also required to access snapshot monitor data.

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.

LOCKWAITS

Information returned:

Table 77. 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 monitor element |
| AGENT_ID | BIGINT | agent_id - Application Handle (agent ID) monitor element |
| APPL_NAME | VARCHAR(256) | appl_name - Application Name monitor element |
| AUTHID | VARCHAR(128) | auth_id - Authorization ID monitor element |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |
| TABSCHEMA | VARCHAR(128) | table_schema - Table Schema Name monitor element |
| TABNAME | VARCHAR(128) | table_name - Table Name monitor element |
| SUBSECTION_NUMBER | BIGINT | ss_number - Subsection Number monitor element |

Table 77. 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 monitor element. 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_WAIT_START_TIME | TIMESTAMP | lock_wait_start_time - Lock Wait Start Timestamp monitor element |
| LOCK_NAME | VARCHAR(32) | lock_name - Lock Name monitor element |

LOCKWAITS

Table 77. Information returned by the LOCKWAITS administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|--------------|--|
| LOCK_MODE | VARCHAR(10) | lock_mode - Lock Mode monitor element. 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 • NX • S • SIX • U • W • X • Z |
| LOCK_MODE_REQUESTED | VARCHAR(10) | lock_mode_requested - Lock Mode Requested monitor element. 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 • NX • S • SIX • U • W • X • Z |
| AGENT_ID_HOLDING_LK | BIGINT | agent_id_holding_lock - Agent ID Holding Lock monitor element |
| APPL_ID_HOLDING_LK | VARCHAR(128) | appl_id_holding_lk - Application ID Holding Lock monitor element |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock Escalation monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information” on page 334
- “SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function – Retrieve lockwait logical data group snapshot information” on page 409
- “Authorization for administrative views” on page 6
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

The schema is SYSIBMADM.

Authorization:

- SELECT or CONTROL privilege on the LOG_UTILIZATION and SNAPDB administrative views.
- SYSMON, SYSCtrl, SYSMaint, or SYSADM authority is also required to access snapshot monitor data.

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 78. 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 monitor element |
| LOG_UTILIZATION_PERCENT | DECIMAL(5,2) | Percent utilization of total log space. |
| TOTAL_LOG_USED_KB | BIGINT | total_log_used - Total Log Space Used monitor element. This interface returns the value in KB. |
| TOTAL_LOG_AVAILABLE_KB | BIGINT | total_log_available - Total Log Available monitor element. This interface returns the value in KB. |

Table 78. Information returned by the LOG_UTILIZATION administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|-----------|--|
| TOTAL_LOG_USED_TOP_KB | BIGINT | tot_log_used_top - Maximum Total Log Space Used monitor element. This interface returns the value in KB. |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group” on page 356
- “Authorization for administrative views” on page 6
- “Database system monitor elements” in *System Monitor Guide and Reference*

LONG_RUNNING_SQL administrative view

The LONG_RUNNING_SQL administrative view returns the longest running SQL statements in the currently connected database.

The schema is SYSIBMADM.

Authorization:

- SELECT or CONTROL privilege on the LONG_RUNNING_SQL, SNAPSTMT, SNAPAPPL_INFO, and SNAPAPPL administrative views.
- SYSMON, SYSCTRL, SYSMAINT, or SYSADM authority is also required to access snapshot monitor data.

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 DB2 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 79. 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. |

Table 79. Information returned by the LONG_RUNNING_SQL administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|--------------|---|
| ELAPSED_TIME_MIN | INTEGER | Elapsed time of the statement in minutes. |
| AGENT_ID | BIGINT | agent_id - Application Handle (agent ID) monitor element |
| APPL_NAME | VARCHAR(256) | appl_name - Application Name monitor element |
| APPL_STATUS | VARCHAR(22) | appl_status - Application Status monitor element. 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 |
| AUTHID | VARCHAR(128) | auth_id - Authorization ID monitor element |
| INBOUND_COMM_ADDRESS | VARCHAR(32) | inbound_comm_address - Inbound Communication Address monitor element |
| STMT_TEXT | CLOB(16 M) | stmt_text - SQL Dynamic Statement Text monitor element |

LONG_RUNNING_SQL

Table 79. Information returned by the LONG_RUNNING_SQL administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------|-----------|--|
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “LOCKWAITS administrative view – Retrieve current lockwaits information” on page 301
- “SNAPSTMT administrative view and SNAP_GET_STMT table function – Retrieve statement snapshot information” on page 415
- “SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information” on page 324
- “SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information” on page 334
- “LOCKS_HELD administrative view – Retrieve information on locks held” on page 297
- “Authorization for administrative views” on page 6
- “Database system monitor elements” in *System Monitor Guide and Reference*

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:

- SELECT or CONTROL privilege on the QUERY_PREP_COST and SNAPDYN_SQL administrative views.
- SYSMON, SYSCTRL, SYSMMAINT, or SYSADM authority is also required to access snapshot monitor data.

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 80. 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. |

QUERY_PREP_COST

Table 80. Information returned by the QUERY_PREP_COST administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|--------------|--|
| NUM_EXECUTIONS | BIGINT | num_executions - Statement Executions monitor element |
| AVERAGE_EXECUTION_TIME_S | BIGINT | Average execution time in seconds. |
| PREP_TIME_MS | BIGINT | prep_time_worst - Statement Worst Preparation Time monitor element |
| PREP_TIME_PERCENT | DECIMAL(5,2) | Percent of execution time spent on preparation. |
| STMT_TEXT | CLOB(2 M) | stmt_text - SQL Dynamic Statement Text monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information” on page 387
- “Authorization for administrative views” on page 6
- “Database system monitor elements” in *System Monitor Guide and Reference*

SNAP_WRITE_FILE procedure

The SNAP_WRITE_FILE procedure writes system snapshot data to a file in the tmp subdirectory of the instance directory.

Syntax:

```
▶▶—SNAP_WRITE_FILE—(—requestType—,—dbName—,—dbpartitionnum—)————▶▶
```

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.

dbpartitionnum

An 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 database partitions. If a null value is specified, -1 is set implicitly.

Authorization:

To execute the procedure, a user must have SYSADM, SYSCTRL, SYSMANT, or SYSMON authority. The saved snapshot can be read by users who do not have SYSADM, SYSCTRL, SYSMANT, or SYSMON authority by passing null values as the inputs to snapshot table functions.

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)
```

SNAP_WRITE_FILE

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.

Related tasks:

- "Capturing database system snapshot information to a file using the SNAP_WRITE_FILE stored procedure" in *System Monitor Guide and Reference*

Related reference:

- "Supported administrative SQL routines and views" on page 8

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.

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 SNAPAGENT_MEMORY_POOL, SNAPAPPL, SNAPAPPL_INFO, SNAPSTMT 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 partitions.

The schema is SYSIBMADM.

Refer to Table 81 on page 317 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPAGENT administrative view and EXECUTE privilege on the SNAP_GET_AGENT table function.

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 partition, aggregate of all database partitions or all database partitions.

SNAPAGENT and SNAP_GET_AGENT

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 partitions.

Refer to Table 81 on page 317 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_AGENT ( ( 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 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 database partitions.

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_AGENT table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_AGENT table function.

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 81. 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 monitor element |
| AGENT_ID | BIGINT | agent_id - Application Handle (agent ID) monitor element |
| AGENT_PID | BIGINT | agent_pid - Process or Thread ID monitor element |
| LOCK_TIMEOUT_VAL | BIGINT | lock_timeout_val - Lock timeout monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data for the row was retrieved. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “GET SNAPSHOT command” in *Command Reference*
- “Database system monitor elements” in *System Monitor Guide and Reference*
- “SNAPSTMT administrative view and SNAP_GET_STMT table function – Retrieve statement snapshot information” on page 415
- “SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function – Retrieve subsection logical monitor group snapshot information” on page 425
- “Administrative views versus table functions” on page 3

SNAPAGENT and SNAP_GET_AGENT

- “SNAP_WRITE_FILE procedure” on page 313
- “SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function – Retrieve memory_pool logical data group snapshot information” on page 319
- “SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information” on page 324
- “SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information” on page 334

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.

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 SNAPAGENT, SNAPAPPL, SNAPAPPL_INFO, SNAPSTMT 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 82 on page 321 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPAGENT_MEMORY_POOL administrative view and EXECUTE privilege on the SNAP_GET_AGENT_MEMORY_POOL table function.

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 partition, aggregate of all database partitions or all database partitions.

Used with the SNAP_GET_AGENT, SNAP_GET_APPL, SNAP_GET_APPL_INFO, SNAP_GET_STMT and

SNAPAGENT_MEMORY_POOL and SNAP_GET_AGENT_MEMORY_POOL

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 82 on page 321 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_AGENT_MEMORY_POOL ( ( 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 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 database partitions.

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_AGENT_MEMORY_POOL table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_AGENT_MEMORY_POOL table function.

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 |

SNAPAGENT_MEMORY_POOL and SNAP_GET_AGENT_MEMORY_POOL

| | | | |
|--------|----|--------------|--------|
| 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 82. 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 monitor element |
| AGENT_ID | BIGINT | agent_id - Application Handle (agent ID) monitor element |
| AGENT_PID | BIGINT | agent_pid - Process or Thread ID monitor element |

SNAPAGENT_MEMORY_POOL and SNAP_GET_AGENT_MEMORY_POOL

Table 82. 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_ID | VARCHAR(14) | pool_id - Memory Pool Identifier monitor element. 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 monitor element |
| POOL_WATERMARK | BIGINT | pool_watermark - Memory Pool Watermark monitor element |
| POOL_CONFIG_SIZE | BIGINT | pool_config_size - Configured Size of Memory Pool monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPAGENT administrative view and SNAP_GET_AGENT table function – Retrieve agent logical data group application snapshot information” on page 315
- “SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information” on page 324

SNAPAGENT_MEMORY_POOL and SNAP_GET_AGENT_MEMORY_POOL

- “SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information” on page 334
- “SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function – Retrieve subsection logical monitor group snapshot information” on page 425
- “SNAPSTMT administrative view and SNAP_GET_STMT table function – Retrieve statement snapshot information” on page 415
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

SNAPAPPL administrative view

This administrative view allows you to retrieve appl logical data group snapshot information for the currently connected database.

Used with the SNAPAGENT, SNAPAGENT_MEMORY_POOL, SNAPAPPL_INFO, SNAPSTMT 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 partitions.

The schema is SYSIBMADM.

Refer to Table 83 on page 326 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPAPPL administrative view and EXECUTE privilege on the SNAP_GET_APPL table function.

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 TABLE 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 partition, aggregate of all database partitions or all database partitions.

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 partitions.

Refer to Table 83 on page 326 for a complete list of information that can be returned.

Syntax:

```

▶▶ SNAP_GET_APPL ( ( 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 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 database partitions.

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_APPL table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_APPL table function.

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

SNAPAPPL and SNAP_GET_APPL

Table 83. 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 monitor element |
| AGENT_ID | BIGINT | agent_id - Application Handle (agent ID) monitor element |
| UOW_LOG_SPACE_USED | BIGINT | uow_log_space_used - Unit of Work Log Space Used monitor element |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| ROWS_WRITTEN | BIGINT | rows_written - Rows Written monitor element |
| INACT_STMTHIST_SZ | BIGINT | inact_stmthist_sz - Statement history list size monitor element |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer Pool Data Writes monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer Pool Index Writes monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer Pool Temporary Data Logical 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_L_READS | BIGINT | pool_temp_index_l_reads - Buffer Pool Temporary Index Logical 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_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads monitor element |

Table 83. 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_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 monitor element |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads 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 |
| DIRECT_READS | BIGINT | direct_reads - Direct Reads From Database monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct Writes to Database 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 |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct Read Time monitor element |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct Write Time monitor element |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread Prefetch Pages monitor element |
| LOCKS_HELD | BIGINT | locks_held - Locks Held monitor element |
| LOCK_WAITS | BIGINT | lock_waits - Lock Waits monitor element |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time Waited On Locks monitor element |
| LOCK_ESCALS | BIGINT | lock_escals - Number of Lock Escalations monitor element |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive Lock Escalations monitor element |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks Detected monitor element |
| TOTAL_SORTS | BIGINT | total_sorts - Total Sorts monitor element |

SNAPAPPL and SNAP_GET_APPL

Table 83. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|-----------|---|
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total Sort Time monitor element |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort Overflows monitor element |
| COMMIT_SQL_STMTS | BIGINT | commit_sql_stmts - Commit Statements Attempted monitor element |
| ROLLBACK_SQL_STMTS | BIGINT | rollback_sql_stmts - Rollback Statements Attempted 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 |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| INT_AUTO_REBINDS | BIGINT | int_auto_rebinds - Internal Automatic Rebinds monitor element |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal Rows Updated monitor element |
| INT_COMMITS | BIGINT | int_commits - Internal Commits monitor element |
| INT_ROLLBACKS | BIGINT | int_rollbacks - Internal Rollbacks monitor element |
| INT_DEADLOCK_ROLLBACKS | BIGINT | int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock monitor element |
| ROWS_DELETED | BIGINT | rows_deleted - Rows Deleted monitor element |
| ROWS_INSERTED | BIGINT | rows_inserted - Rows Inserted monitor element |

Table 83. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|-----------|---|
| ROWS_UPDATED | BIGINT | rows_updated - Rows Updated monitor element |
| ROWS_SELECTED | BIGINT | rows_selected - Rows Selected monitor element |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/Precompiles Attempted monitor element |
| OPEN_REM_CURS | BIGINT | open_rem_curs - Open Remote Cursors monitor element |
| OPEN_REM_CURS_BLK | BIGINT | open_rem_curs_blk - Open Remote Cursors with Blocking monitor element |
| REJ_CURS_BLK | BIGINT | rej_curs_blk - Rejected Block Cursor Requests monitor element |
| ACC_CURS_BLK | BIGINT | acc_curs_blk - Accepted Block Cursor Requests monitor element |
| SQL_REQS_SINCE_COMMIT | BIGINT | sql_reqs_since_commit - SQL Requests Since Last Commit monitor element |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of Lock Timeouts monitor element |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal Rows Inserted monitor element |
| OPEN_LOC_CURS | BIGINT | open_loc_curs - Open Local Cursors monitor element |
| OPEN_LOC_CURS_BLK | BIGINT | open_loc_curs_blk - Open Local Cursors with Blocking monitor element |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package Cache Lookups monitor element |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package Cache Inserts monitor element |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog Cache Lookups monitor element |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog Cache Inserts monitor element |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog Cache Overflows monitor element |
| NUM_AGENTS | BIGINT | num_agents - Number of Agents Working on a Statement monitor element |
| AGENTS_STOLEN | BIGINT | agents_stolen - Stolen Agents monitor element |
| ASSOCIATED_AGENTS_TOP | BIGINT | associated_agents_top - Maximum Number of Associated Agents monitor element |

SNAPAPPL and SNAP_GET_APPL

Table 83. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|--|
| APPL_PRIORITY | BIGINT | appl_priority - Application Agent Priority monitor element |
| APPL_PRIORITY_TYPE | VARCHAR(16) | appl_priority_type - Application Priority Type monitor element. 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 monitor element |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - Section Lookups monitor element |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Section Inserts monitor element |
| LOCKS_WAITING | BIGINT | locks_waiting - Current Agents Waiting On Locks 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 |
| APPL_IDLE_TIME | BIGINT | appl_idle_time - Application Idle Time monitor element |
| UOW_LOCK_WAIT_TIME | BIGINT | uow_lock_wait_time - Total Time Unit of Work Waited on Locks monitor element |
| UOW_COMP_STATUS | VARCHAR(14) | uow_comp_status - Unit of Work Completion Status monitor element. 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 monitor element |

Table 83. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------------|-------------|--|
| AGENT_USR_CPU_TIME_MS | BIGINT | agent_usr_cpu_time - User CPU Time used by Agent monitor element |
| AGENT_SYS_CPU_TIME_S | BIGINT | agent_sys_cpu_time - System CPU Time used by Agent monitor element |
| AGENT_SYS_CPU_TIME_MS | BIGINT | agent_sys_cpu_time - System CPU Time used by Agent monitor element |
| APPL_CON_TIME | TIMESTAMP | appl_con_time - Connection Request Start Timestamp monitor element |
| CONN_COMPLETE_TIME | TIMESTAMP | conn_complete_time - Connection Request Completion Timestamp monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp 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_ELAPSED_TIME_S | BIGINT | uow_elapsed_time - Most Recent Unit of Work Elapsed Time monitor element |
| UOW_ELAPSED_TIME_MS | BIGINT | uow_elapsed_time - Most Recent Unit of Work Elapsed Time monitor element |
| ELAPSED_EXEC_TIME_S | BIGINT | elapsed_exec_time - Statement Execution Elapsed Time monitor element |
| ELAPSED_EXEC_TIME_MS | BIGINT | elapsed_exec_time - Statement Execution Elapsed Time monitor element |
| INBOUND_COMM_ADDRESS | VARCHAR(32) | inbound_comm_address - Inbound Communication Address monitor element |
| LOCK_TIMEOUT_VAL | BIGINT | lock_timeout_val - Lock timeout monitor element |
| PRIV_WORKSPACE_NUM_OVERFLOWS | BIGINT | priv_workspace_num_overflows - Private Workspace Overflows monitor element |
| PRIV_WORKSPACE_SECTION_INSERTS | BIGINT | priv_workspace_section_inserts - Private Workspace Section Inserts monitor element |

SNAPAPPL and SNAP_GET_APPL

Table 83. Information returned by the SNAPAPPL administrative view and the SNAP_GET_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------------|-----------|--|
| PRIV_WORKSPACE_SECTION_LOOKUPS | BIGINT | priv_workspace_section_lookups - Private Workspace Section Lookups monitor element |
| PRIV_WORKSPACE_SIZE_TOP | BIGINT | priv_workspace_size_top - Maximum Private Workspace Size monitor element |
| SHR_WORKSPACE_NUM_OVERFLOWS | BIGINT | shr_workspace_num_overflows - Shared Workspace Overflows monitor element |
| SHR_WORKSPACE_SECTION_INSERTS | BIGINT | shr_workspace_section_inserts - Shared Workspace Section Inserts monitor element |
| SHR_WORKSPACE_SECTION_LOOKUPS | BIGINT | shr_workspace_section_lookups - Shared Workspace Section Lookups monitor element |
| SHR_WORKSPACE_SIZE_TOP | BIGINT | shr_workspace_size_top - Maximum Shared Workspace Size monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data for the row was retrieved. |
| CAT_CACHE_SIZE_TOP | BIGINT | cat_cache_size_top - Catalog Cache High Water Mark monitor element |

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPAGENT administrative view and SNAP_GET_AGENT table function – Retrieve agent logical data group application snapshot information” on page 315
- “SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function – Retrieve memory_pool logical data group snapshot information” on page 319
- “SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information” on page 334
- “SNAPSTMT administrative view and SNAP_GET_STMT table function – Retrieve statement snapshot information” on page 415

- “SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function – Retrieve subsection logical monitor group snapshot information” on page 425
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

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 SNAPAGENT, SNAPAGENT_MEMORY_POOL, SNAPAPPL, SNAPAPPL_INFO, SNAPSTMT 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 partitions.

The schema is SYSIBMADM.

Refer to Table 84 on page 336 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPAPPL_INFO administrative view and EXECUTE privilege on the SNAP_GET_APPL_INFO table function.

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 partition, aggregate of all database partitions or all database partitions.

Used with the SNAP_GET_AGENT, SNAP_GET_AGENT_MEMORY_POOL, SNAP_GET_APPL, SNAP_GET_APPL_INFO, 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 partitions.

Refer to Table 84 on page 336 for a complete list of information that can be returned.

Syntax:

```

▶▶ SNAP_GET_APPL_INFO ( ( 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 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 database partitions.

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_APPL_INFO table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_APPL_INFO table function.

Examples:

Retrieve the status of all applications on the connected database partition.

```

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.

Information returned

SNAPAPPL_INFO and SNAP_GET_APPL_INFO

Table 84. 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) monitor element |
| APPL_STATUS | VARCHAR(22) | appl_status - Application Status monitor element. 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 |
| CODEPAGE_ID | BIGINT | codepage_id - ID of Code Page Used by Application monitor element |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of Associated Agents monitor element |
| COORD_NODE_NUM | SMALLINT | coord_node - Coordinating Node monitor element |

SNAPAPPL_INFO and SNAP_GET_APPL_INFO

Table 84. 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_lvl - User Authorization Level monitor element. This interface returns a text identifier based on the defines in sql.h, and is one or more of the following, separated by ' + ':</p> <ul style="list-style-type: none"> • BINDADD • BINDADD_GRP • CONNECT • CONNECT_GRP • CREATE_EXT_RT • CREATE_EXT_RT_GRP • CREATE_NOT_FENC • CREATE_NOT_FENC_GRP • CREATETAB • CREATETAB_GRP • DBADM • DBADM_GRP • IMPLICIT_SCHEMA • IMPLICIT_SCHEMA_GRP • LOAD • LOAD_GRP • LIBADM • LIBADM_GRP • QUIESCE_CONN • QUIESCE_CONN_GRP • SECADM • SECADM_GRP • SYSADM • SYSADM_GRP • SYSCTRL • SYSCTRL_GRP • SYSMANT • SYSMANT_GRP • SYSMON • SYSMON_GRP • SYSQUIESCE • SYSQUIESCE_GRP |
| CLIENT_PID | BIGINT | client_pid - Client Process ID monitor element |
| COORD_AGENT_PID | BIGINT | coord_agent_pid - Coordinator Agent monitor element |
| STATUS_CHANGE_TIME | TIMESTAMP | status_change_time - Application Status Change Time monitor element |

SNAPAPPL_INFO and SNAP_GET_APPL_INFO

Table 84. 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_PLATFORM | VARCHAR(12) | <p>client_platform - Client Operating Platform monitor element. 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 |

SNAPAPPL_INFO and SNAP_GET_APPL_INFO

Table 84. 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 monitor element. This interface returns a text identifier based on the defines in sqlmon.h, <ul style="list-style-type: none"> • APPC • APPN • CPIC • IPXSPX • LOCAL • NETBIOS • NPIPE • TCPIP (for DB2 UDB) • TCPIP4 • TCPIP6 |
| TERRITORY_CODE | SMALLINT | territory_code - Database Territory Code monitor element |
| APPL_NAME | VARCHAR(256) | appl_name - Application Name monitor element |
| APPL_ID | VARCHAR(128) | appl_id - Application ID monitor element |
| SEQUENCE_NO | VARCHAR(4) | sequence_no - Sequence Number monitor element |
| PRIMARY_AUTH_ID | VARCHAR(128) | auth_id - Authorization ID monitor element |
| SESSION_AUTH_ID | VARCHAR(128) | session_auth_id - Session Authorization ID monitor element |
| CLIENT_NNAME | VARCHAR(128) | client_nname - Configuration NNAME of Client monitor element |
| CLIENT_PRDID | VARCHAR(128) | client_prdid - Client Product/Version ID monitor element |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input Database Alias monitor element |
| CLIENT_DB_ALIAS | VARCHAR(128) | client_db_alias - Database Alias Used by Application monitor element |
| DB_NAME | VARCHAR(128) | db_name - Database Name monitor element |
| DB_PATH | VARCHAR(1024) | db_path - Database Path monitor element |
| EXECUTION_ID | VARCHAR(128) | execution_id - User Login ID monitor element |
| CORR_TOKEN | VARCHAR(128) | corr_token - DRDA Correlation Token monitor element |

SNAPAPPL_INFO and SNAP_GET_APPL_INFO

Table 84. 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 |
|---------------------|--------------|---|
| TPMON_CLIENT_USERID | VARCHAR(256) | tpmon_client_userid - TP Monitor Client User ID monitor element |
| TPMON_CLIENT_WKSTN | VARCHAR(256) | tpmon_client_wkstn - TP Monitor Client Workstation Name monitor element |
| TPMON_CLIENT_APP | VARCHAR(256) | tpmon_client_app - TP Monitor Client Application Name monitor element |
| TPMON_ACC_STR | VARCHAR(200) | tpmon_acc_str - TP Monitor Client Accounting String monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data for the row was retrieved. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPAGENT administrative view and SNAP_GET_AGENT table function – Retrieve agent logical data group application snapshot information” on page 315
- “SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function – Retrieve memory_pool logical data group snapshot information” on page 319
- “SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information” on page 324
- “SNAPSTMT administrative view and SNAP_GET_STMT table function – Retrieve statement snapshot information” on page 415
- “SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function – Retrieve subsection logical monitor group snapshot information” on page 425
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

SNAPBP administrative view

This administrative view allows you to retrieve bufferpool logical group snapshot information for the currently connected database.

Used with the SNAPBP_PART administrative view, 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 85 on page 343 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPBP administrative view and EXECUTE privilege on the SNAP_GET_BP table function.

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 partition, aggregate of all database partitions or all database partitions.

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 85 on page 343 for a complete list of information that can be returned.

Syntax:

SNAPBP and SNAP_GET_BP

```
▶▶ SNAP_GET_BP ( ( 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 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 database partitions.

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_BP table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_BP table function.

Example:

Retrieve total physical and logical reads for all bufferpools for all active databases for the currently connected database partition.

```
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 85. 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 monitor element |
| DB_NAME | VARCHAR(128) | db_name - Database Name monitor element |
| DB_PATH | VARCHAR(1024) | db_path - Database Path monitor element |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input Database Alias monitor element |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer Pool Data Writes monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer Pool Index Writes 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_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 |

SNAPBP and SNAP_GET_BP

Table 85. 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_DATA_READS | BIGINT | pool_async_data_reads - Buffer Pool Asynchronous Data Reads 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_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_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes monitor element |
| 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 |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests monitor element |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests monitor element |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests monitor element |
| DIRECT_READS | BIGINT | direct_reads - Direct Reads From Database monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct Writes to Database 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 |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct Read Time monitor element |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct Write Time monitor element |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread Prefetch Pages monitor element |

Table 85. Information returned by the SNAPBP administrative view and the SNAP_GET_BP table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| FILES_CLOSED | BIGINT | files_closed - Database Files Closed monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer Pool Temporary Data Logical 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_L_READS | BIGINT | pool_temp_index_l_reads - Buffer Pool Temporary Index Logical 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_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads monitor element |
| 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 monitor element |
| PAGES_FROM_BLOCK_IOS | BIGINT | pages_from_block_ios - Total Number of Pages Read by Block IO monitor element |
| PAGES_FROM_VECTORED_IOS | BIGINT | pages_from_vectored_ios - Total Number of Pages Read by Vectored IO monitor element |
| PHYSICAL_PAGE_MAPS | BIGINT | physical_page_maps - Number of Physical Page Maps monitor element |
| VECTORED_IOS | BIGINT | vectored_ios - Number of Vectored IO Requests monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “GET SNAPSHOT command” in *Command Reference*

SNAPBP and SNAP_GET_BP

- “SNAPBP_PART administrative view and SNAP_GET_BP_PART table function – Retrieve bufferpool_nodeinfo logical data group snapshot information” on page 347
- “Administrative views versus table functions” on page 3
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

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 SNAPBP administrative view, 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 86 on page 349 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPBP_PART administrative view and EXECUTE privilege on the SNAP_GET_BP_PART table function.

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 partition, aggregate of all database partitions or all database partitions.

SNAPBP_PART and SNAP_GET_BP_PART

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 86 on page 349 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_BP_PART ( ( 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 for all bufferpools in 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 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 database partitions.

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_BP_PART table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_BP_PART table function.

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 | ... |

SNAPBP_PART and SNAP_GET_BP_PART

```

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                3
...                0                0
...                0                0
...                0                0
...                0                0
...                0                0

```

...

Information returned

Table 86. 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 monitor element |
| DB_NAME | VARCHAR(128) | db_name - Database Name monitor element |
| BP_CUR_BUFFSZ | BIGINT | bp_cur_buffsz - Current Size of Buffer Pool monitor element |
| BP_NEW_BUFFSZ | BIGINT | bp_new_buffsz - New Buffer Pool Size monitor element |
| 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 |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

SNAPBP_PART and SNAP_GET_BP_PART

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPBP administrative view and SNAP_GET_BP table function – Retrieve bufferpool logical group snapshot information” on page 341
- “Database system monitor elements” in *System Monitor Guide and Reference*

SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information

The “SNAPCONTAINER administrative view” and the “SNAP_GET_CONTAINER_V91 table function” on page 352 return table space snapshot information from the tablespace_container logical data group.

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 SNAPTbsp, SNAPTbsp_Part, SNAPTbsp_Quiescer and SNAPTbsp_Range administrative views, 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 87 on page 353 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPCONTAINER administrative view and EXECUTE privilege on the SNAP_GET_CONTAINER_V91 table function.

Example:

Retrieve details for the table space containers for all database partitions 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 | ... |

SNAPCONTAINER and SNAP_GET_CONTAINER_V91

```

... /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_V91 table function

The `SNAP_GET_CONTAINER_V91` 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 partition, aggregate of all database partitions or all database partitions.

Used with the `SNAP_GET_TBSP_V91`, `SNAP_GET_TBSP_PART_V91`, `SNAP_GET_TBSP_QUIESCER` and `SNAP_GET_TBSP_RANGE` table functions, the `SNAP_GET_CONTAINER_V91` table function returns data equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 87 on page 353 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 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 database partitions.

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:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_CONTAINER_V91 table function.

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:\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 87. Information returned by the SNAPCONTAINER administrative view and 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. |

SNAPCONTAINER and SNAP_GET_CONTAINER_V91

Table 87. Information returned by the SNAPCONTAINER administrative view and the SNAP_GET_CONTAINER_V91 table function (continued)

| 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 |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container Name monitor element |
| CONTAINER_ID | BIGINT | container_id - Container Identification monitor element |
| CONTAINER_TYPE | VARCHAR(16) | container_type - Container Type monitor element. 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 monitor element |
| USABLE_PAGES | BIGINT | container_usable_pages - Usable Pages in Container monitor element |
| ACCESSIBLE | SMALLINT | container_accessible - Accessibility of Container monitor element |
| STRIPE_SET | BIGINT | container_stripe_set - Stripe Set monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |
| FS_ID | VARCHAR(22) | fs_id - Unique File System Identification Number monitor element |
| FS_TOTAL_SIZE | BIGINT | fs_total_size - Total Size of a File System monitor element |
| FS_USED_SIZE | BIGINT | fs_used_size - Amount of Space Used on a File System monitor element |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*

SNAPCONTAINER and SNAP_GET_CONTAINER_V91

- “SNAPTBSP administrative view and SNAP_GET_TBSP_V91 table function – Retrieve tablespace logical data group snapshot information” on page 441
- “SNAPTBSP QUIESCER administrative view and SNAP_GET_TBSP QUIESCER table function – Retrieve quiescer table space snapshot information” on page 452
- “SNAPTBSP_RANGE administrative view and SNAP_GET_TBSP_RANGE table function – Retrieve range snapshot information” on page 456
- “SNAPTBSP_PART administrative view and SNAP_GET_TBSP_PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information” on page 447
- “Database system monitor elements” in *System Monitor Guide and Reference*

SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group

The “SNAPDB administrative view” and the “SNAP_GET_DB_V91 table function” return snapshot information from the database (dbase) logical group.

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 the SNAPDB_MEMORY_POOL, SNAPDETAILOG, SNAPHADR and SNAPSTORAGE_PATHS administrative views, 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 88 on page 358 for a complete list of information that is returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPDB administrative view and EXECUTE privilege on the SNAP_GET_DB_V91 table function.

Example:

Retrieve the status, platform, location, and connect time for all database partitions 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 |

SNAP_GET_DB_V91 table function

The SNAP_GET_DB_V91 table function returns the same information as the SNAPDB administrative view.

Used in conjunction with the SNAP_GET_DB_MEMORY_POOL, SNAP_GET_DETAILLOG_V91, SNAP_GET_HADR and SNAP_GET_STORAGE_PATHS table functions, the SNAP_GET_DB_V91

table function provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES CLP** command.

Refer to Table 88 on page 358 for a complete list of information that is returned.

Syntax:

```
▶▶ SNAP_GET_DB_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 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 database partitions.

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_DB_V91 table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_DB_V91 table function.

Examples:

Example 1: Retrieve the status, platform, location, and connect time as an aggregate view across all database partitions 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_V91('', -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.

SNAPDB and SNAP_GET_DB_V91

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 partitions 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_V91(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
```

SNAPDB administrative view and SNAP_GET_DB_V91 table function metadata

Table 88. Information returned by the SNAPDB administrative view and SNAP_GET_DB_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 monitor element |
| DB_PATH | VARCHAR(1024) | db_path - Database Path monitor element |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input Database Alias monitor element |
| DB_STATUS | VARCHAR(12) | db_status - Status of Database monitor element. 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 |
| CATALOG_PARTITION | SMALLINT | catalog_node - Catalog Node Number monitor element |
| CATALOG_PARTITION_NAME | VARCHAR(128) | catalog_node_name - Catalog Node Network Name monitor element |

Table 88. Information returned by the SNAPDB administrative view and SNAP_GET_DB_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------|-------------|---|
| SERVER_PLATFORM | VARCHAR(12) | server_platform - Server Operating System monitor element. 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 • WINDOWS95 |
| DB_LOCATION | VARCHAR(12) | db_location - Database Location monitor element. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • LOCAL • REMOTE |

Table 88. Information returned by the SNAPDB administrative view and SNAP_GET_DB_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|------------------|---|
| DB_CONN_TIME | TIMESTAMP | db_conn_time - Database Activation Timestamp monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |
| 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 |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum Number of Coordinating Agents monitor element |
| LOCKS_HELD | BIGINT | locks_held - Locks Held monitor element |
| LOCK_WAITS | BIGINT | lock_waits - Lock Waits monitor element |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time Waited On Locks monitor element |
| LOCK_LIST_IN_USE | BIGINT | lock_list_in_use - Total Lock List Memory In Use monitor element |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks Detected monitor element |
| LOCK_ESCALS | BIGINT | lock_escals - Number of Lock Escalations monitor element |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive Lock Escalations monitor element |
| LOCKS_WAITING | BIGINT | locks_waiting - Current Agents Waiting On Locks monitor element |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of Lock Timeouts monitor element |

Table 88. Information returned by the SNAPDB administrative view and SNAP_GET_DB_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|--|
| NUM_INDOUBT_TRANS | BIGINT | num_indoubt_trans - Number of Indoubt Transactions monitor element |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total Sort Heap Allocated 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 Water Mark monitor element |
| POST_SHRTHRESHOLD_SORTS | BIGINT | post_shrthreshold_sorts - Post threshold sorts monitor element |
| TOTAL_SORTS | BIGINT | total_sorts - Total Sorts monitor element |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total Sort Time monitor element |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort Overflows monitor element |
| ACTIVE_SORTS | BIGINT | active_sorts - Active Sorts monitor element |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads monitor element |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads monitor element |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer Pool Asynchronous Data Reads monitor element |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer Pool Data Writes monitor element |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer Pool Asynchronous Data Writes monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |

SNAPDB and SNAP_GET_DB_V91

Table 88. Information returned by the SNAPDB administrative view and SNAP_GET_DB_V91 table function (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 monitor element |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads monitor element |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer Pool Asynchronous Index Reads monitor element |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer Pool Index Writes monitor element |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer Pool Asynchronous Index Writes monitor element |
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads monitor element |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads monitor element |
| POOL_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes monitor element |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads monitor element |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes monitor element |
| 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 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 |
| 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 88. Information returned by the SNAPDB administrative view and SNAP_GET_DB_V91 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 monitor element |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests monitor element |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests monitor element |
| 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 |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time Waited for Prefetch monitor element |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread Prefetch Pages monitor element |
| DIRECT_READS | BIGINT | direct_reads - Direct Reads From Database monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct Writes to Database 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 |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct Read Time monitor element |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct Write Time monitor element |
| FILES_CLOSED | BIGINT | files_closed - Database Files Closed monitor element |
| ELAPSED_EXEC_TIME_S | BIGINT | elapsed_exec_time - Statement Execution Elapsed Time monitor element |
| ELAPSED_EXEC_TIME_MS | BIGINT | elapsed_exec_time - Statement Execution Elapsed Time monitor element |
| COMMIT_SQL_STMTS | BIGINT | commit_sql_stmts - Commit Statements Attempted monitor element |

SNAPDB and SNAP_GET_DB_V91

Table 88. Information returned by the SNAPDB administrative view and SNAP_GET_DB_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|-----------|---|
| ROLLBACK_SQL_STMTS | BIGINT | rollback_sql_stmts - Rollback Statements Attempted 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 |
| INT_AUTO_REBINDS | BIGINT | int_auto_rebinds - Internal Automatic Rebinds 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 |
| INT_COMMITS | BIGINT | int_commits - Internal Commits monitor element |
| INT_ROLLBACKS | BIGINT | int_rollbacks - Internal Rollbacks monitor element |
| INT_DEADLOCK_ROLLBACKS | BIGINT | int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock 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 |
| ROWS_SELECTED | BIGINT | rows_selected - Rows Selected monitor element |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |

Table 88. Information returned by the SNAPDB administrative view and SNAP_GET_DB_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|-----------|--|
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/Precompiles Attempted 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_S | BIGINT | log_read_time - Log Read Time monitor element |
| LOG_READ_TIME_NS | BIGINT | log_read_time - Log Read Time monitor element |
| LOG_WRITES | BIGINT | log_writes - Number of Log Pages Written monitor element |
| LOG_WRITE_TIME_S | BIGINT | log_write_time - Log Write Time monitor element |
| LOG_WRITE_TIME_NS | 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 Full Log Buffers monitor element |
| NUM_LOG_DATA_FOUND_IN_BUFFER | BIGINT | num_log_data_found_in_buffer - Number of Log Data Found In Buffer monitor element |
| APPL_ID_OLDEST_XACT | BIGINT | appl_id_oldest_xact - Application with Oldest Transaction monitor element |
| LOG_TO_REDO_FOR_RECOVERY | BIGINT | log_to_redo_for_recovery - Amount of Log to be Redone for Recovery monitor element |

Table 88. Information returned by the SNAPDB administrative view and SNAP_GET_DB_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------------|-----------|--|
| LOG_HELD_BY_DIRTY_PAGES | BIGINT | log_held_by_dirty_pages - Amount of Log Space Accounted for by Dirty Pages monitor element |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package Cache Lookups monitor element |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package Cache Inserts monitor element |
| PKG_CACHE_NUM_OVERFLOWS | BIGINT | pkg_cache_num_overflows - Package Cache Overflows monitor element |
| PKG_CACHE_SIZE_TOP | BIGINT | pkg_cache_size_top - Package Cache High Water Mark monitor element |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - Section Lookups monitor element |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Section Inserts monitor element |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog Cache Lookups monitor element |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog Cache Inserts monitor element |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog Cache Overflows monitor element |
| CAT_CACHE_SIZE_TOP | BIGINT | cat_cache_size_top - Catalog Cache High Water Mark monitor element |
| PRIV_WORKSPACE_SIZE_TOP | BIGINT | priv_workspace_size_top - Maximum Private Workspace Size monitor element |
| PRIV_WORKSPACE_NUM_OVERFLOWS | BIGINT | priv_workspace_num_overflows - Private Workspace Overflows monitor element |
| PRIV_WORKSPACE_SECTION_INSERTS | BIGINT | priv_workspace_section_inserts - Private Workspace Section Inserts monitor element |
| PRIV_WORKSPACE_SECTION_LOOKUPS | BIGINT | priv_workspace_section_lookups - Private Workspace Section Lookups monitor element |
| SHR_WORKSPACE_SIZE_TOP | BIGINT | shr_workspace_size_top - Maximum Shared Workspace Size monitor element |
| SHR_WORKSPACE_NUM_OVERFLOWS | BIGINT | shr_workspace_num_overflows - Shared Workspace Overflows monitor element |
| SHR_WORKSPACE_SECTION_INSERTS | BIGINT | shr_workspace_section_inserts - Shared Workspace Section Inserts monitor element |

Table 88. Information returned by the SNAPDB administrative view and SNAP_GET_DB_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------------|-----------|--|
| SHR_WORKSPACE_SECTION_LOOKUPS | BIGINT | shr_workspace_section_lookups - Shared Workspace Section Lookups 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_OVERFLOWES | BIGINT | hash_join_overflows - Hash Join Overflows monitor element |
| HASH_JOIN_SMALL_OVERFLOWES | 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 |
| ACTIVE_HASH_JOINS | BIGINT | active_hash_joins - Active hash joins monitor element |
| NUM_DB_STORAGE_PATHS | BIGINT | num_db_storage_paths - Number of automatic storage paths monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |
| SMALLEST_LOG_AVAIL_NODE | INTEGER | smallest_log_avail_node - Node with Least Available Log Space monitor element |

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPDB_MEMORY_POOL administrative view and SNAP_GET_DB_MEMORY_POOL table function – Retrieve database level memory usage information” on page 369
- “SNAPHADR administrative view and SNAP_GET_HADR table function – Retrieve hadr logical data group snapshot information” on page 398
- “SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG_V91 table function – Retrieve snapshot information from the detail_log logical data group” on page 383

SNAPDB and SNAP_GET_DB_V91

- “SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS table function – Retrieve automatic storage path information” on page 421
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

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 SNAPDB, SNAPDETAILLOG, SNAPHADR and SNAPSTORAGE_PATHS administrative views, 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 89 on page 371 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPDB_MEMORY_POOL administrative view and EXECUTE privilege on the SNAP_GET_DB_MEMORY_POOL table function.

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 partition, aggregate of all database partitions or all database partitions.

SNAPDB_MEMORY_POOL and SNAP_GET_DB_MEMORY_POOL

Used with the SNAP_GET_DB_V91, SNAP_GET_DETAILLOG_V91, SNAP_GET_HADR and SNAP_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 89 on page 371 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_DB_MEMORY_POOL ( ( 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 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 database partitions.

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_DB_MEMORY_POOL table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_DB_MEMORY_POOL table function.

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.

SNAPDB_MEMORY_POOL and SNAP_GET_DB_MEMORY_POOL

| 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 89. 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 monitor element |

SNAPDB_MEMORY_POOL and SNAP_GET_DB_MEMORY_POOL

Table 89. 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 monitor element. 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 monitor element |
| POOL_CUR_SIZE | BIGINT | pool_cur_size - Current Size of Memory Pool monitor element |
| POOL_WATERMARK | BIGINT | pool_watermark - Memory Pool Watermark monitor element |
| POOL_CONFIG_SIZE | BIGINT | pool_config_size - Configured Size of Memory Pool monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*

SNAPDB_MEMORY_POOL and SNAP_GET_DB_MEMORY_POOL

- “SNAPHADR administrative view and SNAP_GET_HADR table function – Retrieve hadr logical data group snapshot information” on page 398
- “SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group” on page 356
- “SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG_V91 table function – Retrieve snapshot information from the detail_log logical data group” on page 383
- “SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS table function – Retrieve automatic storage path information” on page 421
- “Database system monitor elements” in *System Monitor Guide and Reference*

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 DB2 database manager (dbm) logical grouping information.

SNAPDBM administrative view

Used with the SNAPDBM_MEMORY_POOL, SNAPFCM, SNAPFCM_PART and SNAPSWITCHES administrative views, the SNAPDBM administrative view provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

The schema is SYSIBMADM.

Refer to Table 90 on page 375 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPDBM administrative view and EXECUTE privilege on the SNAP_GET_DBM table function.

Example:

Retrieve database manager status and connection information for all database partitions.

```
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 partition, aggregate of all database partitions or all database partitions.

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 90 for a complete list of information that can be returned.

Syntax:

```

▶▶ SNAP_GET_DBM ( [ dbpartitionnum ] ) ▶▶
    
```

The schema is SYSPROC.

Table function parameter:

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 database partitions. If this input option is not used, data will be returned from all database partitions.

If *dbpartitionnum* 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:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_DBM table function.

Example:

Retrieve the start time and current status of database partition 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 90. 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 monitor element |
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post Threshold Sorts monitor element |
| PIPED_SORTS_REQUESTED | BIGINT | pipedsorts_requested - Piped Sorts Requested monitor element |
| PIPED_SORTS_ACCEPTED | BIGINT | pipedsorts_accepted - Piped Sorts Accepted monitor element |

SNAPDBM and SNAP_GET_DBM

Table 90. Information returned by the SNAPDBM administrative view and the SNAP_GET_DBM table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|---|
| REM_CONS_IN | BIGINT | rem_cons_in - Remote Connections To Database Manager monitor element |
| 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 monitor element |
| LOCAL_CONS_IN_EXEC | BIGINT | local_cons_in_exec - Local Connections Executing in the Database Manager monitor element |
| CON_LOCAL_DBASES | BIGINT | con_local_databases - Local Databases with Current Connects monitor element |
| AGENTS_REGISTERED | BIGINT | agents_registered - Agents Registered monitor element |
| AGENTS_WAITING_ON_TOKEN | BIGINT | agents_waiting_on_token - Agents Waiting for a Token monitor element |
| DB2_STATUS | VARCHAR(12) | db2_status - Status of DB2 Instance monitor element. 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 monitor element |
| AGENTS_WAITING_TOP | BIGINT | agents_waiting_top - Maximum Number of Agents Waiting monitor element |
| COMM_PRIVATE_MEM | BIGINT | comm_private_mem - Committed Private Memory 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 |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum Number of Coordinating Agents monitor element |
| MAX_AGENT_OVERFLOW | BIGINT | max_agent_overflows - Maximum Agent Overflows monitor element |

Table 90. 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_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 |
| POST_THRESHOLD_HASH_JOINS | BIGINT | post_threshold_hash_joins - Hash Join Threshold monitor element |
| NUM_GW_CONN_SWITCHES | BIGINT | num_gw_conn_switches - Connection Switches monitor element |
| DB2START_TIME | TIMESTAMP | db2start_time - Start Database Manager Timestamp monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |
| NUM_NODES_IN_DB2_INSTANCE | INTEGER | num_nodes_in_db2_instance - Number of Nodes in Partition monitor element |
| PRODUCT_NAME | VARCHAR(32) | product_name - Product Name monitor element |
| SERVICE_LEVEL | VARCHAR(18) | service_level - Service Level monitor element |
| SORT_HEAP_TOP | BIGINT | sort_heap_top - Sort Private Heap High Water Mark monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “GET SNAPSHOT command” in *Command Reference*
- “Administrative views versus table functions” on page 3

SNAPDBM and SNAP_GET_DBM

- “SNAPDBM_MEMORY_POOL administrative view and SNAP_GET_DBM_MEMORY_POOL table function – Retrieve database manager level memory usage information” on page 379
- “SNAPFCM administrative view and SNAP_GET_FCM table function – Retrieve the fcm logical data group snapshot information” on page 392
- “SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function – Retrieve the fcm_node logical data group snapshot information” on page 395
- “SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function – Retrieve database snapshot switch state information” on page 429
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

SNAPDBM_MEMORY_POOL administrative view

Used with the SNAPDBM, SNAPFCM, SNAPFCM_PART and SNAPSWITCHES administrative views, the SNAPDBM_MEMORY_POOL administrative view provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

The schema is SYSIBMADM.

Refer to Table 91 on page 380 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPDBM_MEMORY_POOL administrative view and EXECUTE privilege on the SNAP_GET_DBM_MEMORY_POOL table function.

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 partition, aggregate of all database partitions or all database partitions.

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 91 on page 380 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_DBM_MEMORY_POOL ( ( dbpartitionnum ) ) ▶▶
```

SNAPDBM_MEMORY_POOL and SNAP_GET_DBM_MEMORY_POOL

The schema is SYSPROC.

Table function parameter:

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 database partitions. If this input option is not used, data will be returned from all database partitions.

If *dbpartitionnum* 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 partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_DBM_MEMORY_POOL table function.

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 91. 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. |

SNAPDBM_MEMORY_POOL and SNAP_GET_DBM_MEMORY_POOL

Table 91. 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 monitor element. 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 monitor element |
| POOL_WATERMARK | BIGINT | pool_watermark - Memory Pool Watermark monitor element |
| POOL_CONFIG_SIZE | BIGINT | pool_config_size - Configured Size of Memory Pool monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPDBM administrative view and SNAP_GET_DBM table function – Retrieve the dbm logical grouping snapshot information” on page 374
- “SNAPFCM administrative view and SNAP_GET_FCM table function – Retrieve the fcm logical data group snapshot information” on page 392

SNAPDBM_MEMORY_POOL and SNAP_GET_DBM_MEMORY_POOL

- “SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function – Retrieve the fcm_node logical data group snapshot information” on page 395
- “SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function – Retrieve database snapshot switch state information” on page 429
- “Database system monitor elements” in *System Monitor Guide and Reference*

SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG_V91 table function – Retrieve snapshot information from the detail_log logical data group

The “SNAPDETAILLOG administrative view” and the “SNAP_GET_DETAILLOG_V91 table function” return snapshot information from the detail_log logical data group.

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 the SNAPDB, SNAPDB_MEMORY_POOL, SNAPHADR and SNAPSTORAGE_PATHS administrative views, 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 92 on page 385 for a complete list of information that is returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPDETAILLOG administrative view and EXECUTE privilege on the SNAP_GET_DETAILLOG_V91 table function.

Example:

Retrieve log information for all database partitions 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_V91 table function

The SNAP_GET_DETAILLOG_V91 table function returns the same information as the SNAPDETAILLOG administrative view.

Used in conjunction with the SNAP_GET_DB_V91, SNAP_GET_DB_MEMORY_POOL, SNAP_GET_HADR and

SNAPDETAILLOG and SNAP_GET_DETAILLOG_V91

SNAP_GET_STORAGE_PATHS table functions, the SNAP_GET_DETAILLOG table function provides information equivalent to the GET SNAPSHOT FOR ALL DATABASES CLP command.

Refer to Table 92 on page 385 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 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 database partitions.

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:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_DETAILLOG_V91 table function.

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.

SNAPDETAILLOG and SNAP_GET_DETAILLOG_V91

```

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          -
...
...

```

SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG_V91 table function metadata

Table 92. Information returned by the SNAPDETAILLOG administrative view and 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 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 |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPHADR administrative view and SNAP_GET_HADR table function – Retrieve hadr logical data group snapshot information” on page 398
- “SNAPDB_MEMORY_POOL administrative view and SNAP_GET_DB_MEMORY_POOL table function – Retrieve database level memory usage information” on page 369
- “SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group” on page 356

SNAPDETAILLOG and SNAP_GET_DETAILLOG_V91

- “SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS table function – Retrieve automatic storage path information” on page 421
- “Database system monitor elements” in *System Monitor Guide and Reference*

SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information

The “SNAPDYN_SQL administrative view” and the “SNAP_GET_DYN_SQL_V91 table function” on page 388 return snapshot information from the dynsql logical data group.

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 93 on page 389 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPDYN_SQL administrative view and EXECUTE privilege on the SNAP_GET_DYN_SQL_V91 table function.

Example:

Retrieve a list of dynamic SQL run on all database partitions 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, 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 ...
```

SNAPDYN_SQL and SNAP_GET_DYN_SQL_V91

```
... 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_V91 table function

The SNAP_GET_DYN_SQL_V91 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 partition, aggregate of all database partitions or all database partitions.

This table function returns information equivalent to the **GET SNAPSHOT FOR DYNAMIC SQL ON database-alias** CLP command.

Refer to Table 93 on page 389 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_DYN_SQL_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 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 database partitions.

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_DYN_SQL_V91 table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_DYN_SQL_V91 table function.

Example:

Retrieve a list of dynamic SQL run on the currently connected database partition 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_V91('','-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,
```

Information returned

Table 93. Information returned by the SNAPDYN_SQL administrative view and the SNAP_GET_DYN_SQL_V91 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 monitor element |
| NUM_COMPILATIONS | BIGINT | num_compilations - Statement Compilations monitor element |

SNAPDYN_SQL and SNAP_GET_DYN_SQL_V91

Table 93. Information returned by the SNAPDYN_SQL administrative view and the SNAP_GET_DYN_SQL_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|--|
| PREP_TIME_WORST | BIGINT | prep_time_worst - Statement Worst Preparation Time monitor element |
| PREP_TIME_BEST | BIGINT | prep_time_best - Statement Best Preparation Time 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 |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| ROWS_WRITTEN | BIGINT | rows_written - Rows Written monitor element |
| STMT_SORTS | BIGINT | stmt_sorts - Statement Sorts monitor element |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort Overflows monitor element |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total Sort Time monitor element |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads monitor element |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads monitor element |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads monitor element |
| POOL_XDA_L_READS | BIGINT | pool_xda_l_reads - Buffer Pool XDA Data Logical Reads monitor element |

Table 93. Information returned by the SNAPDYN_SQL administrative view and the SNAP_GET_DYN_SQL_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|-----------|---|
| POOL_XDA_P_READS | BIGINT | pool_xda_p_reads - Buffer Pool XDA Data Physical Reads monitor element |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads monitor element |
| 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 monitor element |
| TOTAL_EXEC_TIME_MS | BIGINT | total_exec_time - Elapsed Statement Execution Time monitor element |
| TOTAL_USR_CPU_TIME | BIGINT | total_usr_cpu_time - Total User CPU for a Statement monitor element |
| TOTAL_USR_CPU_TIME_MS | BIGINT | total_usr_cpu_time - Total User CPU for a Statement monitor element |
| TOTAL_SYS_CPU_TIME | BIGINT | total_sys_cpu_time - Total System CPU for a Statement monitor element |
| TOTAL_SYS_CPU_TIME_MS | BIGINT | total_sys_cpu_time - Total System CPU for a Statement monitor element |
| STMT_TEXT | CLOB(2 M) | stmt_text - SQL Dynamic Statement Text monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

SNAPFCM administrative view

Used with the SNAPDBM, SNAPDBM_MEMORY_POOL, SNAPFCM_PART and SNAPSWITCHES administrative views, the SNAPFCM administrative view provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

The schema is SYSIBMADM.

Refer to Table 94 on page 393 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPFCM administrative view and EXECUTE privilege on the SNAP_GET_FCM table function.

Example:

Retrieve information about the fast communication manager’s message buffers on all database partitions.

```
SELECT BUFF_FREE, BUFF_FREE_BOTTOM, DBPARTITIONNUM
FROM SYSIBMADM.SNAPFCM ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| BUFF_FREE | BUFF_FREE_BOTTOM | DBPARTITIONNUM |
|-----------|------------------|----------------|
| 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 partition, aggregate of all database partitions or all database partitions.

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 94 on page 393 for a complete list of information that can be returned.

Syntax:

```
→ SNAP_GET_FCM ( [ dbpartitionnum ] ) →
```

The schema is SYSPROC.

Table function parameter:

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 database partitions. If this input option is not used, data will be returned from all database partitions

If *dbpartitionnum* 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 partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_FCM table function.

Example:

Retrieve information about the fast communication manager’s message buffers on database partition 1.

```
SELECT BUFF_FREE, BUFF_FREE_BOTTOM, DBPARTITIONNUM
FROM TABLE(SYSPROC.SNAP_GET_FCM( 1 )) AS T
```

The following is an example of output from this query.

| BUFF_FREE | BUFF_FREE_BOTTOM | DBPARTITIONNUM |
|-----------|------------------|----------------|
| 5120 | 5100 | 1 |

Information returned

Table 94. 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 monitor element |
| BUFF_FREE_BOTTOM | BIGINT | buff_free_bottom - Minimum FCM Buffers Free monitor element |
| CH_FREE | BIGINT | ch_free - Channels Currently Free monitor element |
| CH_FREE_BOTTOM | BIGINT | ch_free_bottom - Minimum Channels Free monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

SNAPFCM and SNAP_GET_FCM

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “GET SNAPSHOT command” in *Command Reference*
- “Administrative views versus table functions” on page 3
- “SNAPDBM_MEMORY_POOL administrative view and SNAP_GET_DBM_MEMORY_POOL table function – Retrieve database manager level memory usage information” on page 379
- “SNAPDBM administrative view and SNAP_GET_DBM table function – Retrieve the dbm logical grouping snapshot information” on page 374
- “SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function – Retrieve the fcm_node logical data group snapshot information” on page 395
- “SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function – Retrieve database snapshot switch state information” on page 429
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

SNAPFCM_PART administrative view

Used with the SNAPDBM, SNAPDBM_MEMORY_POOL, SNAPFCM and SNAPSWITCHES administrative views, the SNAPFCM_PART administrative view provides the data equivalent to the **GET SNAPSHOT FOR DBM** command.

The schema is SYSIBMADM.

Refer to Table 95 on page 396 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPFCM_PART administrative view and EXECUTE privilege on the SNAP_GET_FCM_PART table function.

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 DBPARTITIONNUM = 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 partition, aggregate of all database partitions or all database partitions.

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 95 on page 396 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_FCM_PART ( [ dbpartitionnum ] ) ▶▶
```

SNAPFCM_PART and SNAP_GET_FCM_PART

The schema is SYSPROC.

Table function parameter:

dbpartitionnum

An optional input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current partition, or -2 for an aggregate of all partitions. If this input option is not used, data will be returned from all partitions

If *dbpartitionnum* 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 database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_FCM_PART table function.

Example:

Retrieve buffers sent and received information for the fast communication manager for all database partitions.

```
SELECT FCM_DBPARTITIONNUM, TOTAL_BUFFERS_SENT, TOTAL_BUFFERS_RCVD,
       DBPARTITIONNUM FROM TABLE(SNAP_GET_FCM_PART()) AS T
ORDER BY DBPARTITIONNUM
```

The following is an example of output from this query.

| FCM_DBPARTITIONNUM | TOTAL_BUFFERS_SENT | TOTAL_BUFFERS_RCVD | DBPARTITIONNUM |
|--------------------|--------------------|--------------------|----------------|
| 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 95. 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 monitor element. 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 |

SNAPFCM_PART and SNAP_GET_FCM_PART

Table 95. 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 monitor element |
| TOTAL_BUFFERS_RCVD | BIGINT | total_buffers_rcvd - Total FCM Buffers Received monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |
| 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). |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “GET SNAPSHOT command” in *Command Reference*
- “Administrative views versus table functions” on page 3
- “SNAPDBM_MEMORY_POOL administrative view and SNAP_GET_DBM_MEMORY_POOL table function – Retrieve database manager level memory usage information” on page 379
- “SNAPDBM administrative view and SNAP_GET_DBM table function – Retrieve the dbm logical grouping snapshot information” on page 374
- “SNAPFCM administrative view and SNAP_GET_FCM table function – Retrieve the fcm logical data group snapshot information” on page 392
- “SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function – Retrieve database snapshot switch state information” on page 429
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

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 SNAPDB, SNAPDB_MEMORY_POOL, SNAPDETAILLOG and SNAPSTORAGE_PATHS administrative views, 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 96 on page 400 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPHADR administrative view and EXECUTE privilege on the SNAP_GET_HADR table function.

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 partition, aggregate of all database partitions or all database partitions.

Used with the SNAP_GET_DB_V91, SNAP_GET_DB_MEMORY_POOL, SNAP_GET_DETAILLOG_V91 and SNAP_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 96 on page 400 for a complete list of information that can be returned.

Syntax:

```

▶▶ SNAP_GET_HADR ( ( 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 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 database partitions.

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_HADR table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_HADR table function.

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.

SNAPHADR and SNAP_GET_HADR

Information returned

Table 96. 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 monitor element |
| HADR_ROLE | VARCHAR(10) | hadr_role - HADR Role monitor element. 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 monitor element. 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 monitor element. 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 • SYNC |
| HADR_CONNECT_STATUS | VARCHAR(12) | hadr_connect_status - HADR Connection Status monitor element. 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 monitor element |
| HADR_HEARTBEAT | INTEGER | hadr_heartbeat - HADR Heartbeat monitor element |
| HADR_LOCAL_HOST | VARCHAR(255) | hadr_local_host - HADR Local Host monitor element |
| HADR_LOCAL_SERVICE | VARCHAR(40) | hadr_local_service - HADR Local Service monitor element |

Table 96. 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_REMOTE_HOST | VARCHAR(255) | hadr_remote_host - HADR Remote Host monitor element |
| HADR_REMOTE_SERVICE | VARCHAR(40) | hadr_remote_service - HADR Remote Service monitor element |
| HADR_REMOTE_INSTANCE | VARCHAR(128) | hadr_remote_instance - HADR Remote Instance monitor element |
| HADR_TIMEOUT | BIGINT | hadr_timeout - HADR Timeout monitor element |
| HADR_PRIMARY_LOG_FILE | VARCHAR(255) | hadr_primary_log_file - HADR Primary Log File monitor element |
| HADR_PRIMARY_LOG_PAGE | BIGINT | hadr_primary_log_page - HADR Primary Log Page monitor element |
| HADR_PRIMARY_LOG_LSN | BIGINT | hadr_primary_log_lsn - HADR Primary Log LSN monitor element |
| HADR_STANDBY_LOG_FILE | VARCHAR(255) | hadr_standby_log_file - HADR Standby Log File monitor element |
| HADR_STANDBY_LOG_PAGE | BIGINT | hadr_standby_log_page - HADR Standby Log Page monitor element |
| HADR_STANDBY_LOG_LSN | BIGINT | hadr_standby_log_lsn - HADR Standby Log LSN monitor element |
| HADR_LOG_GAP | BIGINT | hadr_log_gap - HADR Log Gap monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Administrative views versus table functions” on page 3
- “SNAP_WRITE_FILE procedure” on page 313
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPDB_MEMORY_POOL administrative view and SNAP_GET_DB_MEMORY_POOL table function – Retrieve database level memory usage information” on page 369
- “SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group” on page 356
- “SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG_V91 table function – Retrieve snapshot information from the detail_log logical data group” on page 383
- “SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS table function – Retrieve automatic storage path information” on page 421

SNAPHADR and SNAP_GET_HADR

- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

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 97 on page 404 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPLOCK administrative view and EXECUTE privilege on the SNAP_GET_LOCK table function.

Example:

Retrieve lock information for the database partition 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 partition, aggregate of all database partitions or all database partitions.

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 97 on page 404 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_LOCK ( ( dbname [ , dbpartitionnum ] ) ) ▶▶
```

SNAPLOCK and SNAP_GET_LOCK

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.

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 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 database partitions.

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_LOCK table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_LOCK table function.

Example:

Retrieve lock information for the current database partition of the currently connected database.

```
SELECT AGENT_ID, LOCK_OBJECT_TYPE, LOCK_MODE, LOCK_STATUS
FROM TABLE(SNAP_GET_LOCK('',-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 97. 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) monitor element |

SNAPLOCK and SNAP_GET_LOCK

Table 97. Information returned by the SNAPLOCK administrative view and the SNAP_GET_LOCK table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------|-------------|--|
| TAB_FILE_ID | BIGINT | table_file_id - Table File ID monitor element |
| LOCK_OBJECT_TYPE | VARCHAR(18) | lock_object_type - Lock Object Type Waited On monitor element. 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 |

SNAPLOCK and SNAP_GET_LOCK

Table 97. 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 monitor element. 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 • NX • S • SIX • U • W • X • Z |
| LOCK_STATUS | VARCHAR(10) | lock_status - Lock Status monitor element. 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 monitor element |
| TABNAME | VARCHAR(128) | table_name - Table Name monitor element |
| TABSCHEMA | VARCHAR(128) | table_schema - Table Schema Name monitor element |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |

Table 97. 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_ATTRIBUTES | VARCHAR(128) | lock_attributes - Lock Attributes monitor element. 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 monitor element |
| LOCK_CURRENT_MODE | VARCHAR(10) | lock_current_mode - Original Lock Mode Before Conversion monitor element. 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 • NX • S • SIX • U • W • X • Z |
| LOCK_HOLD_COUNT | BIGINT | lock_hold_count - Lock Hold Count monitor element |
| LOCK_NAME | VARCHAR(32) | lock_name - Lock Name monitor element |
| LOCK_RELEASE_FLAGS | BIGINT | lock_release_flags - Lock Release Flags monitor element |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition Identifier monitor element. For a non-partitioned table, this element is NULL. |

SNAPLOCK and SNAP_GET_LOCK

Table 97. Information returned by the SNAPLOCK administrative view and the SNAP_GET_LOCK table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------|-----------|--|
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “GET SNAPSHOT command” in *Command Reference*
- “Administrative views versus table functions” on page 3
- “SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function – Retrieve lockwait logical data group snapshot information” on page 409
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

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 98 on page 411 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPLOCKWAIT administrative view and EXECUTE privilege on the SNAP_GET_LOCKWAIT table function.

Example:

Retrieve lock wait information on database partition 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 partition, aggregate of all database partitions or all database partitions.

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.

SNAPLOCKWAIT and SNAP_GET_LOCKWAIT

Refer to Table 98 on page 411 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_LOCKWAIT (—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 a null value 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 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 database partitions.

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_LOCKWAIT table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_LOCKWAIT table function.

Example:

Retrieve lock wait information on current database partition 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
```

SNAPLOCKWAIT and SNAP_GET_LOCKWAIT

Information returned

Table 98. 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) monitor element |
| SUBSECTION_NUMBER | BIGINT | ss_number - Subsection Number monitor element |
| LOCK_MODE | VARCHAR(10) | lock_mode - Lock Mode monitor element. 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• NX• S• SIX• U• W• X• Z |

SNAPLOCKWAIT and SNAP_GET_LOCKWAIT

Table 98. 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_OBJECT_TYPE | VARCHAR(18) | lock_object_type - Lock Object Type Waited On monitor element. 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 monitor element |
| LOCK_WAIT_START_TIME | TIMESTAMP | lock_wait_start_time - Lock Wait Start Timestamp monitor element |

SNAPLOCKWAIT and SNAP_GET_LOCKWAIT

Table 98. 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_REQUESTED | VARCHAR(10) | lock_mode_requested - Lock Mode Requested monitor element. 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 • NX • S • SIX • U • W • X • Z |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock Escalation monitor element |
| TABNAME | VARCHAR(128) | table_name - Table Name monitor element |
| TABSCHEMA | VARCHAR(128) | table_schema - Table Schema Name monitor element |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |
| APPL_ID_HOLDING_LK | VARCHAR(128) | appl_id_holding_lk - Application ID Holding Lock monitor element |
| LOCK_ATTRIBUTES | VARCHAR(128) | lock_attributes - Lock Attributes monitor element. 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 |

SNAPLOCKWAIT and SNAP_GET_LOCKWAIT

Table 98. 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 monitor element. 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• NX• S• SIX• U• W• X• Z |
| LOCK_NAME | VARCHAR(32) | lock_name - Lock Name monitor element |
| LOCK_RELEASE_FLAGS | BIGINT | lock_release_flags - Lock Release Flags monitor element. |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition Identifier monitor element. For a non-partitioned table, this element is NULL. |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPLOCK administrative view and SNAP_GET_LOCK table function – Retrieve lock logical data group snapshot information” on page 403
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

SNAPSTMT administrative view

This administrative view allows you to retrieve statement snapshot information for the currently connected database.

Used with the SNAPAGENT, SNAPAGENT_MEMORY_POOL, SNAPAPPL, SNAPAPPL_INFO 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 partitions.

The schema is SYSIBMADM.

Refer to Table 99 on page 417 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPSTMT administrative view and EXECUTE privilege on the SNAP_GET_STMT table function.

Example:

Retrieve rows read, written and operation performed for statements executed on the currently connected single-partition 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 partition, aggregate of all database partitions or all database partitions.

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 99 on page 417 for a complete list of information that can be returned.

Syntax:

SNAPSTMT and SNAP_GET_STMT

```

▶▶ SNAP_GET_STMT ( ( 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 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 database partitions.

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_STMT table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_STMT table function.

Example:

Retrieve rows read, written and operation performed for statements executed on current database partition 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 99. 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 monitor element |
| AGENT_ID | BIGINT | agent_id - Application Handle (agent ID) monitor element |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| ROWS_WRITTEN | BIGINT | rows_written - Rows Written 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 |
| STMT_TYPE | VARCHAR(20) | stmt_type - Statement Type monitor element. 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 |

SNAPSTMT and SNAP_GET_STMT

Table 99. 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 monitor element. 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 monitor element |
| QUERY_COST_ESTIMATE | BIGINT | query_cost_estimate - Query Cost Estimate monitor element |
| QUERY_CARD_ESTIMATE | BIGINT | query_card_estimate - Query Number of Rows Estimate monitor element |
| DEGREE_PARALLELISM | BIGINT | degree_parallelism - Degree of Parallelism monitor element |
| STMT_SORTS | BIGINT | stmt_sorts - Statement Sorts monitor element |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total Sort Time monitor element |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort Overflows monitor element |

Table 99. 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_DELETED | BIGINT | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal Rows Updated monitor element |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal Rows Inserted monitor element |
| FETCH_COUNT | BIGINT | fetch_count - Number of Successful Fetches monitor element |
| STMT_START | TIMESTAMP | stmt_start - Statement Operation Start Timestamp monitor element |
| STMT_STOP | TIMESTAMP | stmt_stop - Statement Operation Stop Timestamp monitor element |
| STMT_USR_CPU_TIME_S | BIGINT | stmt_usr_cpu_time - User CPU Time used by Statement monitor element |
| STMT_USR_CPU_TIME_MS | BIGINT | stmt_usr_cpu_time - User CPU Time used by Statement monitor element |
| STMT_SYS_CPU_TIME_S | BIGINT | stmt_sys_cpu_time - System CPU Time used by Statement monitor element |
| STMT_SYS_CPU_TIME_MS | BIGINT | stmt_sys_cpu_time - System CPU Time used by Statement monitor element |
| STMT_ELAPSED_TIME_S | BIGINT | stmt_elapsed_time - Most Recent Statement Elapsed Time monitor element |
| STMT_ELAPSED_TIME_MS | BIGINT | stmt_elapsed_time - Most Recent Statement Elapsed Time monitor element |
| BLOCKING_CURSOR | SMALLINT | blocking_cursor - Blocking Cursor monitor element |
| STMT_NODE_NUMBER | SMALLINT | stmt_node_number - Statement Node monitor element |
| CURSOR_NAME | VARCHAR(128) | cursor_name - Cursor Name monitor element |
| CREATOR | VARCHAR(128) | creator - Application Creator monitor element |
| PACKAGE_NAME | VARCHAR(128) | package_name - Package Name monitor element |
| STMT_TEXT | CLOB(16 M) | stmt_text - SQL Dynamic Statement Text monitor element |
| CONSISTENCY_TOKEN | VARCHAR(128) | consistency_token - Package Consistency Token monitor element |
| PACKAGE_VERSION_ID | VARCHAR(128) | package_version_id - Package Version monitor element |

SNAPSTMT and SNAP_GET_STMT

Table 99. 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_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |
| POOL_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads monitor element |
| POOL_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer Pool Temporary Data Logical 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_L_READS | BIGINT | pool_temp_index_l_reads - Buffer Pool Temporary Index Logical 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_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads monitor element |
| POOL_TEMP_XDA_P_READS | BIGINT | pool_temp_xda_p_reads - Buffer Pool Temporary XDA Data Physical Reads monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8

- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function – Retrieve subsection logical monitor group snapshot information” on page 425
- “SNAPAGENT administrative view and SNAP_GET_AGENT table function – Retrieve agent logical data group application snapshot information” on page 315
- “SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function – Retrieve memory_pool logical data group snapshot information” on page 319
- “SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information” on page 324
- “SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information” on page 334
- “Database system monitor elements” in *System Monitor Guide and Reference*

SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS table function – Retrieve automatic storage path information

The “SNAPSTORAGE_PATHS administrative view” and the “SNAP_GET_STORAGE_PATHS table function” on page 422 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.

SNAPSTORAGE_PATHS administrative view

This administrative view allows you to retrieve automatic storage path information for the currently connected database.

Used with the SNAPDB, SNAPDETAILLOG, SNAPHADR and SNAPDB_MEMORY_POOL administrative views, 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 100 on page 423 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPSTORAGE_PATHS administrative view and EXECUTE privilege on the SNAP_GET_STORAGE_PATHS table function.

Example:

Retrieve the storage path for the currently connected single-partition 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
```


SNAPSTORAGE_PATHS and SNAP_GET_STORAGE_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 table function

The SNAP_GET_STORAGE_PATHS table function returns the same information as the SNAPSTORAGE_PATHS 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 with the SNAP_GET_DB_V91, SNAP_GET_DETAILLOG_V91, SNAP_GET_HADR and SNAP_GET_DB_MEMORY_POOL table functions, the SNAP_GET_STORAGE_PATHS table function provides information equivalent to the **GET SNAPSHOT FOR ALL DATABASES CLP** command.

Refer to Table 100 on page 423 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_STORAGE_PATHS ( ( 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 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 database partitions.

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_STORAGE_PATHS table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority

SNAPSTORAGE_PATHS and SNAP_GET_STORAGE_PATHS

- EXECUTE privilege on the SNAP_GET_STORAGE_PATHS table function.

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(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/jessicae/sdb
MYDB     /home/jessicae/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 100. Information returned by the SNAPSTORAGE_PATHS administrative view and the SNAP_GET_STORAGE_PATHS 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 monitor element |
| DB_STORAGE_PATH | VARCHAR(256) | db_storage_path - Automatic storage path monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |
| FS_ID | VARCHAR(22) | fs_id - Unique File System Identification Number monitor element |
| FS_TOTAL_SIZE | BIGINT | fs_total_size - Total Size of a File System monitor element |
| FS_USED_SIZE | BIGINT | fs_used_size - Amount of Space Used on a File System monitor element |
| STO_PATH_FREE_SIZE | BIGINT | sto_path_free_sz - Automatic Storage Path Free Space monitor element |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3

SNAPSTORAGE_PATHS and SNAP_GET_STORAGE_PATHS

- “GET SNAPSHOT command” in *Command Reference*
- “SNAPHADR administrative view and SNAP_GET_HADR table function – Retrieve hadr logical data group snapshot information” on page 398
- “SNAPDB_MEMORY_POOL administrative view and SNAP_GET_DB_MEMORY_POOL table function – Retrieve database level memory usage information” on page 369
- “SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group” on page 356
- “SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG_V91 table function – Retrieve snapshot information from the detail_log logical data group” on page 383
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

SNAPSUBSECTION administrative view

This administrative view allows you to retrieve subsection logical monitor group snapshot information for the currently connected database.

Used with the SNAPAGENT, SNAPAGENT_MEMORY_POOL, SNAPAPPL, SNAPAPPL_INFO and SNAPSTMT administrative views, the SNAPSUBSECTION administrative view provides information equivalent to the **GET SNAPSHOT FOR APPLICATIONS on database-alias** CLP command, but retrieves data from all database partitions.

The schema is SYSIBMADM.

Refer to Table 101 on page 426 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPSUBSECTION administrative view and EXECUTE privilege on the SNAP_GET_SUBSECTION table function.

Example:

Get status for subsections executing on all database partitions.

```
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 partition, aggregate of all database partitions or all database partitions.

Refer to Table 101 on page 426 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 partitions.

Syntax:

SNAPSUBSECTION and SNAP_GET_SUBSECTION

```

▶▶ SNAP_GET_SUBSECTION ( ( 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 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 database partitions.

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_SUBSECTION table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_SUBSECTION table function.

Example:

Get status for subsections executing on all database partitions.

```

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 101. 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. |

SNAPSUBSECTION and SNAP_GET_SUBSECTION

Table 101. Information returned by the SNAPSUBSECTION administrative view and the SNAP_GET_SUBSECTION table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| DB_NAME | VARCHAR(128) | db_name - Database Name monitor element |
| STMT_TEXT | CLOB(16 M) | stmt_text - SQL Dynamic Statement Text monitor element |
| SS_EXEC_TIME | BIGINT | ss_exec_time - Subsection Execution Elapsed Time monitor element |
| TQ_TOT_SEND_SPILLS | BIGINT | tq_tot_send_spills - Total Number of Tablequeue Buffers Overflowed monitor element |
| TQ_CUR_SEND_SPILLS | BIGINT | tq_cur_send_spills - Current Number of Tablequeue Buffers Overflowed monitor element |
| TQ_MAX_SEND_SPILLS | BIGINT | tq_max_send_spills - Maximum Number of Tablequeue Buffers Overflows monitor element |
| TQ_ROWS_READ | BIGINT | tq_rows_read - Number of Rows Read from Tablequeues monitor element |
| TQ_ROWS_WRITTEN | BIGINT | tq_rows_written - Number of Rows Written to Tablequeues monitor element |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| ROWS_WRITTEN | BIGINT | rows_written - Rows Written monitor element |
| SS_USR_CPU_TIME_S | BIGINT | ss_usr_cpu_time - User CPU Time used by Subsection monitor element |
| SS_USR_CPU_TIME_MS | BIGINT | ss_usr_cpu_time - User CPU Time used by Subsection monitor element |
| SS_SYS_CPU_TIME_S | BIGINT | ss_sys_cpu_time - System CPU Time used by Subsection monitor element |
| SS_SYS_CPU_TIME_MS | BIGINT | ss_sys_cpu_time - System CPU Time used by Subsection monitor element |
| SS_NUMBER | INTEGER | ss_number - Subsection Number monitor element |
| SS_STATUS | VARCHAR(20) | ss_status - Subsection Status monitor element. 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 |

SNAPSUBSECTION and SNAP_GET_SUBSECTION

Table 101. 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_NODE_NUMBER | SMALLINT | ss_node_number - Subsection Node Number monitor element |
| TQ_NODE_WAITED_FOR | SMALLINT | tq_node_waited_for - Waited for Node on a Tablequeue monitor element |
| TQ_WAIT_FOR_ANY | INTEGER | tq_wait_for_any - Waiting for Any Node to Send on a Tablequeue monitor element |
| TQ_ID_WAITING_ON | INTEGER | tq_id_waiting_on - Waited on Node on a Tablequeue monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPAGENT administrative view and SNAP_GET_AGENT table function – Retrieve agent logical data group application snapshot information” on page 315
- “SNAPAGENT_MEMORY_POOL administrative view and SNAP_GET_AGENT_MEMORY_POOL table function – Retrieve memory_pool logical data group snapshot information” on page 319
- “SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information” on page 324
- “SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information” on page 334
- “SNAPSTMT administrative view and SNAP_GET_STMT table function – Retrieve statement snapshot information” on page 415
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

SNAPSWITCHES administrative view

This view provides the data equivalent to the **GET DBM MONITOR SWITCHES** CLP command.

The schema is SYSIBMADM.

Refer to Table 102 on page 430 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPSWITCHES administrative view and EXECUTE privilege on the SNAP_GET_SWITCHES table function.

Example:

Retrieve DBM monitor switches state information for all database partitions.

```
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 partition, aggregate of all database partitions or all database partitions.

This table function provides the data equivalent to the **GET DBM MONITOR SWITCHES** CLP command.

Refer to Table 102 on page 430 for a complete list of information that can be returned.

Syntax:

SNAPSWITCHES and SNAP_GET_SWITCHES

▶▶ SNAP_GET_SWITCHES ([*dbpartitionnum*]) ▶▶

The schema is SYSPROC.

Table function parameter:

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 database partitions. If this input option is not used, data will be returned from all database partitions.

If *dbpartitionnum* 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 partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_SWITCHES table function.

Examples:

Retrieve DBM monitor switches state information for the current database partition.

```
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 102. 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. |

SNAPSWITCHES and SNAP_GET_SWITCHES

Table 102. Information returned by the SNAPSWITCHES administrative view and the SNAP_GET_SWITCHES table function (continued)

| Column name | Data type | Description |
|--------------------|-----------|---|
| 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). |
| 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 | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET DATABASE MANAGER MONITOR SWITCHES command” in *Command Reference*
- “Database system monitor elements” in *System Monitor Guide and Reference*

SNAPTAB administrative view and SNAP_GET_TAB_V91 table function – Retrieve table logical data group snapshot information

The “SNAPTAB administrative view” and the “SNAP_GET_TAB_V91 table function” return snapshot information from the table logical data group.

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 103 on page 433 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPTAB administrative view and EXECUTE privilege on the SNAP_GET_TAB_V91 table function.

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_V91 table function

The SNAP_GET_TAB_V91 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 partition, aggregate of all database partitions or all database partitions.

Used in conjunction with the SNAP_GET_TAB_REORG table function, the SNAP_GET_TAB_V91 table function returns equivalent information to the **GET SNAPSHOT FOR TABLES ON database-alias** CLP command.

Refer to Table 103 on page 433 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 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 database partitions.

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:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_TAB_V91 table function.

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 103. Information returned by the SNAPTAB administrative view and 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 monitor element |
| TABNAME | VARCHAR(128) | table_name - Table Name monitor element |

SNAPTAB and SNAP_GET_TAB_V91

Table 103. Information returned by the SNAPTAB administrative view and the SNAP_GET_TAB_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-------------|--|
| TAB_FILE_ID | BIGINT | table_file_id - Table File ID monitor element |
| TAB_TYPE | VARCHAR(14) | table_type - Table Type monitor element. 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 monitor element |
| INDEX_OBJECT_PAGES | BIGINT | index_object_pages - Index Object Pages monitor element |
| LOB_OBJECT_PAGES | BIGINT | lob_object_pages - LOB Object Pages monitor element |
| LONG_OBJECT_PAGES | BIGINT | long_object_pages - Long Object Pages monitor element |
| XDA_OBJECT_PAGES | BIGINT | xda_object_pages - XDA Object Pages monitor element |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| ROWS_WRITTEN | BIGINT | rows_written - Rows Written monitor element |
| OVERFLOW_ACCESSES | BIGINT | overflow_accesses - Accesses to Overflowed Records monitor element |
| PAGE_REORGS | BIGINT | page_reorgs - Page Reorganizations monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |
| TBSP_ID | BIGINT | tablespace_id - Table Space Identification monitor element |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition Identifier monitor element. For a non-partitioned table, this element will be NULL. |

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPTAB_REORG administrative view and SNAP_GET_TAB_REORG table function – Retrieve table reorganization snapshot information” on page 436
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

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 104 on page 438 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPTAB_REORG administrative view and EXECUTE privilege on the SNAP_GET_TAB_REORG table function.

Example:

Select details on reorganization operations for all database partitions on the currently connected database.

```
SELECT SUBSTR(TABNAME, 1, 15) AS TAB_NAME, SUBSTR(TABSHEMA, 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 |

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 partition, aggregate of all database partitions or all database partitions.

SNAPTAB_REORG and SNAP_GET_TAB_REORG

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 104 on page 438 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_TAB_REORG ( ( 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 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 database partitions.

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_REORG table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_TAB_REORG table function.

Example:

Select details on reorganization operations for database partition 1 on the currently connected database.

```
SELECT SUBSTR(TABNAME, 1, 15) AS TAB_NAME, SUBSTR(TABSHEMA, 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.

SNAPTAB_REORG and SNAP_GET_TAB_REORG

Output from this query (continued).

```

... REORG_STATUS REORG_COMPLETION DBPARTITIONNUM
... -----
... COMPLETED SUCCESS 1
...

```

Information returned

Table 104. 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 monitor element |
| TABSCHEMA | VARCHAR (128) | table_schema - Table Schema Name monitor element |
| PAGE_REORGS | BIGINT | page_reorgs - Page Reorganizations monitor element |
| REORG_PHASE | VARCHAR (16) | reorg_phase - Reorganize Phase monitor element. This interface returns a text identifier based on defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • BUILD • DICT_SAMPLE • INDEX_RECREATE • REPLACE • SORT or SORT+DICT_SAMPLE. |
| REORG_MAX_PHASE | INTEGER | reorg_max_phase - Maximum Reorganize Phase monitor element |
| REORG_CURRENT_COUNTER | BIGINT | reorg_current_counter - Reorganize Progress monitor element |
| REORG_MAX_COUNTER | BIGINT | reorg_max_counter - Total Amount of Reorganization monitor element |

SNAPTAB_REORG and SNAP_GET_TAB_REORG

Table 104. 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 monitor element. 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 <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, one of:</p> <ul style="list-style-type: none"> • +INDEXSCAN or +TABLESCAN • +LONGLOB or +DATAONLY <p>If offline, and option is specified, any of:</p> <ul style="list-style-type: none"> • +CHOOSE_TEMP • +KEEP_DICTIONARY • +RESET_DICTIONARY <p>If online, and option is specified:</p> <ul style="list-style-type: none"> • +NOTRUNCATE <p>For example, if a REORG TABLE TEST.EMPLOYEE was run, the following would be displayed:</p> <p>RECLAIM+OFFLINE+ALOW_READ +TABLESCAN+DATAONLY</p> |
| REORG_STATUS | VARCHAR (10) | <p>reorg_status - Table Reorganize Status monitor element. This interface returns a text identifier based on defines in sqlmon.h and is one of:</p> <ul style="list-style-type: none"> • COMPLETED • PAUSED • STARTED • STOPPED • TRUNCATE |
| REORG_COMPLETION | VARCHAR (10) | <p>reorg_completion - Reorganization Completion Flag monitor element. This interface returns a text identifier, based on defines in sqlmon.h and is one of:</p> <ul style="list-style-type: none"> • FAIL • SUCCESS |

SNAPTAB_REORG and SNAP_GET_TAB_REORG

Table 104. 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_START | TIMESTAMP | reorg_start - Table Reorganize Start Time monitor element |
| REORG_END | TIMESTAMP | reorg_end - Table Reorganize End Time monitor element |
| REORG_PHASE_START | TIMESTAMP | reorg_phase_start - Reorganize Phase Start Time monitor element |
| REORG_INDEX_ID | BIGINT | reorg_index_id - Index Used to Reorganize the Table monitor element |
| REORG_TBSPC_ID | BIGINT | reorg_tbspc_id - Table Space Where Table or Data partition is Reorganized monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |
| DATA_PARTITION_ID | INTEGER | data_partition_id - Data Partition Identifier monitor element. For a non-partitioned table, this element will be NULL. |
| REORG_ROWSCOMPRESSED | BIGINT | reorg_rows_compressed - Rows Compressed monitor element |
| REORG_ROWSREJECTED | BIGINT | reorg_rows_rejected_for_compression - Rows Rejected for Compression monitor element |
| REORG_LONG_TBSPC_ID | BIGINT | reorg_long_tbspc_id - Table Space Where Long Objects are Reorganized monitor element |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPTAB administrative view and SNAP_GET_TAB_V91 table function – Retrieve table logical data group snapshot information” on page 432
- “Database system monitor elements” in *System Monitor Guide and Reference*

SNAPTbsp administrative view and SNAP_GET_TBSP_V91 table function – Retrieve tablespace logical data group snapshot information

The “SNAPTbsp administrative view” and the “SNAP_GET_TBSP_V91 table function” return snapshot information from the tablespace logical data group.

SNAPTbsp administrative view

This administrative view allows you to retrieve tablespace 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 105 on page 443 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPTbsp administrative view and EXECUTE privilege on the SNAP_GET_TBSP_V91 table function.

Example:

Retrieve a list of table spaces on the catalog database partition 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_V91 table function

The SNAP_GET_TBSP_V91 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 partition, aggregate of all database partitions or all database partitions.

Used in conjunction with the SNAP_GET_TBSP_PART_V91, SNAP_GET_TBSP_QUIESCER, SNAP_GET_TBSP_RANGE, SNAP_GET_CONTAINER_V91 table functions, the SNAP_GET_TBSP_V91 table function returns information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 105 on page 443 for a complete list of information that can be returned.

Syntax:

SNAPTbsp and SNAP_GET_TBSP_V91

```
▶▶ 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 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 database partitions.

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:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_TBSP_V91 table function.

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 105. Information returned by the SNAPTbsp administrative view and 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 monitor element |
| TBSP_ID | BIGINT | tablespace_id - Table Space Identification monitor element |
| TBSP_TYPE | VARCHAR(10) | tablespace_type - Table Space Type monitor element. 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 monitor element. 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 monitor element |
| TBSP_EXTENT_SIZE | BIGINT | tablespace_extent_size - Table Space Extent Size monitor element |
| TBSP_PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table Space Prefetch Size monitor element |
| TBSP_CUR_POOL_ID | BIGINT | tablespace_cur_pool_id - Buffer Pool Currently Being Used monitor element |
| TBSP_NEXT_POOL_ID | BIGINT | tablespace_next_pool_id - Buffer Pool That Will Be Used at Next Startup monitor element |
| FS_CACHING | SMALLINT | fs_caching - File System Caching monitor element |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads monitor element |

SNAPTbsp and SNAP_GET_TBSP_V91

Table 105. Information returned by the SNAPTbsp administrative view and the SNAP_GET_TBSP_V91 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 monitor element |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer Pool Asynchronous Data Reads monitor element |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer Pool Data Writes monitor element |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer Pool Asynchronous Data Writes monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads monitor element |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads monitor element |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer Pool Asynchronous Index Reads monitor element |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer Pool Index Writes monitor element |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer Pool Asynchronous Index Writes 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_XDA_WRITES | BIGINT | pool_xda_writes - Buffer Pool XDA Data Writes monitor element |
| POOL_ASYNC_XDA_READS | BIGINT | pool_async_xda_reads - Buffer Pool Asynchronous XDA Data Reads monitor element |
| POOL_ASYNC_XDA_WRITES | BIGINT | pool_async_xda_writes - Buffer Pool Asynchronous XDA Data Writes monitor element |
| POOL_TEMP_XDA_L_READS | BIGINT | pool_temp_xda_l_reads - Buffer Pool Temporary XDA Data Logical Reads monitor element |

Table 105. Information returned by the SNAPTbsp administrative view and the SNAP_GET_TBSP_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|---|
| 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 monitor element |
| POOL_WRITE_TIME | BIGINT | pool_write_time - Total Buffer Pool Physical Write Time monitor element |
| 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 |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests monitor element |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests monitor element |
| POOL_ASYNC_XDA_READ_REQS | BIGINT | pool_async_xda_read_reqs - Buffer Pool Asynchronous XDA Read Requests monitor element |
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer Pool No Victim Buffers monitor element |
| DIRECT_READS | BIGINT | direct_reads - Direct Reads From Database monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct Writes to Database 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 |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct Read Time monitor element |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct Write Time monitor element |
| FILES_CLOSED | BIGINT | files_closed - Database Files Closed monitor element |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread Prefetch Pages monitor element |

SNAPTBSP and SNAP_GET_TBSP_V91

Table 105. Information returned by the SNAPTBSP administrative view and the SNAP_GET_TBSP_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|-------------|--|
| TBSP_REBALANCER_MODE | VARCHAR(10) | tablespace_rebalancer_mode - Rebalancer Mode monitor element. 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 - Using automatic storage monitor element |
| TBSP_AUTO_RESIZE_ENABLED | SMALLINT | tablespace_auto_resize_enabled - Auto-resize enabled monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related concepts:

- “XML storage object overview” in *Administration Guide: Planning*

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPTBSP QUIESCER administrative view and SNAP_GET_TBSP QUIESCER table function – Retrieve quiescer table space snapshot information” on page 452
- “SNAPTBSP RANGE administrative view and SNAP_GET_TBSP RANGE table function – Retrieve range snapshot information” on page 456
- “SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information” on page 351
- “SNAPTBSP PART administrative view and SNAP_GET_TBSP PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information” on page 447
- “Database system monitor elements” in *System Monitor Guide and Reference*

SNAPTbsp_PART administrative view and SNAP_GET_TBSP_PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information

The “SNAPTbsp_PART administrative view” and the “SNAP_GET_TBSP_PART_V91 table function” return snapshot information from the tablespace_nodeinfo logical data group.

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 106 on page 449 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPTbsp_PART administrative view and EXECUTE privilege on the SNAP_GET_TBSP_PART_V91 table function.

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_V91 table function

The SNAP_GET_TBSP_PART_V91 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_V91, SNAP_GET_TBSP_QUIESCER, SNAP_GET_TBSP_RANGE, SNAP_GET_CONTAINER_V91 table functions, the

SNAPTbsp_part and SNAP_Get_Tbsp_part_V91

SNAP_Get_Tbsp_part_V91 table function returns information equivalent to the **GET SNAPSHOT FOR TABLESPACES ON database-alias** CLP command.

Refer to Table 106 on page 449 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_Get_Tbsp_part_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 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 database partitions.

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_part_V91 table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_Get_Tbsp_part_V91 table function.

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_V91(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 |

SNAPTbsp_PART and SNAP_GET_Tbsp_PART_V91

SYSTOOLSPACE
SYSTOOLSTMPSPACE

3 NORMAL
4 NORMAL

5 record(s) selected.

Information returned

Table 106. Information returned by the SNAPTbsp_PART administrative view and the SNAP_GET_Tbsp_PART_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 monitor element |
| Tbsp_ID | BIGINT | tablespace_id - Table Space Identification monitor element |
| Tbsp_STATE | VARCHAR (256) | tablespace_state - Table Space State monitor element. This interface returns a text identifier based on defines in sqlutil.h and is combination of the following separated by a '+' sign: <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 monitor element |
| Tbsp_NUM_QUIESCERS | BIGINT | tablespace_num_quiescers - Number of Quiescers monitor element |

SNAPTbsp_part and SNAP_Get_Tbsp_part_V91

Table 106. Information returned by the SNAPTbsp_part administrative view and the SNAP_Get_Tbsp_part_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|--------------|--|
| Tbsp_State_Change_Object_ID | BIGINT | tablespace_state_change_object_id - State Change Object Identification monitor element |
| Tbsp_State_Change_Tbsp_ID | BIGINT | tablespace_state_change_ts_id - State Change Table Space Identification monitor element |
| Tbsp_Min_Recovery_Time | TIMESTAMP | tablespace_min_recovery_time - Minimum Recovery Time For Rollforward monitor element |
| Tbsp_Total_Pages | BIGINT | tablespace_total_pages - Total Pages in Table Space monitor element |
| Tbsp_Usable_Pages | BIGINT | tablespace_usable_pages - Usable Pages in Table Space monitor element |
| Tbsp_Used_Pages | BIGINT | tablespace_used_pages - Used Pages in Table Space monitor element |
| Tbsp_Free_Pages | BIGINT | tablespace_free_pages - Free Pages in Table Space monitor element |
| Tbsp_Pending_Free_Pages | BIGINT | tablespace_pending_free_pages - Pending Free Pages in Table Space monitor element |
| Tbsp_Page_Top | BIGINT | tablespace_page_top - Table Space High Water Mark monitor element |
| Rebalancer_Mode | VARCHAR (10) | tablespace_rebalancer_mode - Rebalancer Mode monitor element. This interface returns a text identifier based on defines in sqlmon.h, and is one of: <ul style="list-style-type: none"> • FWD_REBAL • NO_REBAL • REV_REBAL |
| Rebalancer_Extents_Remaining | BIGINT | tablespace_rebalancer_extents_remaining - Total Number of Extents to be Processed by the Rebalancer monitor element |
| Rebalancer_Extents_Processed | BIGINT | tablespace_rebalancer_extents_processed - Number of Extents the Rebalancer has Processed monitor element |
| Rebalancer_Priority | BIGINT | tablespace_rebalancer_priority - Current Rebalancer Priority monitor element |
| Rebalancer_Start_Time | TIMESTAMP | tablespace_rebalancer_start_time - Rebalancer Start Time monitor element |
| Rebalancer_Restart_Time | TIMESTAMP | tablespace_rebalancer_restart_time - Rebalancer Restart Time monitor element |
| Rebalancer_Last_Extent_Moved | BIGINT | tablespace_rebalancer_last_extent_moved - Last Extent Moved by the Rebalancer monitor element |
| Tbsp_Num_Ranges | BIGINT | tablespace_num_ranges - Number of Ranges in the Table Space Map monitor element |
| Tbsp_Num_Containers | BIGINT | tablespace_num_containers - Number of Containers in Table Space monitor element |

Table 106. Information returned by the SNAPTbsp_PART administrative view and the SNAP_GET_Tbsp_PART_V91 table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|--|
| Tbsp_INITIAL_SIZE | BIGINT | tablespace_initial_size - Initial table space size monitor element |
| Tbsp_CURRENT_SIZE | BIGINT | tablespace_current_size - Current table space size monitor element |
| Tbsp_MAX_SIZE | BIGINT | tablespace_max_size - Maximum table space size monitor element |
| Tbsp_INCREASE_SIZE | BIGINT | tablespace_increase_size - Increase size in bytes monitor element |
| Tbsp_INCREASE_SIZE_PERCENT | SMALLINT | tablespace_increase_size_percent - Increase size by percent monitor element |
| Tbsp_LAST_RESIZE_TIME | TIMESTAMP | tablespace_last_resize_time - Time of last successful resize monitor element |
| Tbsp_LAST_RESIZE_FAILED | SMALLINT | tablespace_last_resize_failed - Last resize attempt failed monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPTbsp administrative view and SNAP_GET_Tbsp_V91 table function – Retrieve tablespace logical data group snapshot information” on page 441
- “SNAPTbsp QUIESCER administrative view and SNAP_GET_Tbsp QUIESCER table function – Retrieve quiescer table space snapshot information” on page 452
- “SNAPTbsp RANGE administrative view and SNAP_GET_Tbsp RANGE table function – Retrieve range snapshot information” on page 456
- “SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information” on page 351
- “Database system monitor elements” in *System Monitor Guide and Reference*

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.

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, SNAPTbsp_PART, SNAPTbsp_RANGE, SNAPCONTAINER administrative views, 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 107 on page 454 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPTbsp_QUIESCER administrative view and EXECUTE privilege on the SNAP_GET_TBSP_QUIESCER table function.

Example:

Retrieve information on quiesced table spaces for all database partitions 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 |
| ... | | |

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 partition, aggregate of all database partitions or all database partitions.

SNAPTbsp_QUIESCER and SNAP_GET_Tbsp_QUIESCER

Used with the SNAP_GET_Tbsp_V91, SNAP_GET_Tbsp_PART_V91, SNAP_GET_Tbsp_RANGE, SNAP_GET_CONTAINER_V91 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 107 on page 454 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_Tbsp_QUIESCER ( ( 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 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 database partitions.

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_QUIESCER table function takes a snapshot for the currently connected database and database partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_Tbsp_QUIESCER table function.

Example:

Retrieve information on quiesced table spaces for database partition 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.

SNAPTbsp_QUIESCER and SNAP_GET_TBSP_QUIESCER

```

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 107. 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 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_agent_id | BIGINT | quiescer_agent_id - Quiescer Agent Identification monitor element |
| Quiescer_state | VARCHAR(14) | quiescer_state - Quiescer State monitor element. 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 | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Database system monitor elements” in *System Monitor Guide and Reference*
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*

SNAPTBSP QUIESCER and SNAP_GET_TBSP QUIESCER

- “SNAPTBSP administrative view and SNAP_GET_TBSP_V91 table function – Retrieve tablespace logical data group snapshot information” on page 441
- “SNAPTBSP_PART administrative view and SNAP_GET_TBSP_PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information” on page 447
- “SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information” on page 351
- “SNAPTBSP_RANGE administrative view and SNAP_GET_TBSP_RANGE table function – Retrieve range snapshot information” on page 456

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” on page 457 return information from a range snapshot.

SNAPTbsp_range administrative view

This administrative view allows you to retrieve range snapshot information for the currently connected database.

Used with the SNAPTbsp, SNAPTbsp_Part, SNAPTbsp_Quiescer and SNAPCONTAINER administrative views, 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 108 on page 458 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPTbsp_range administrative view and EXECUTE privilege on the Snap_Get_Tbsp_range table function.

Example:

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 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 | ... |

SNAPTbsp_RANGE and SNAP_GET_Tbsp_RANGE

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 partition, aggregate of all database partitions or all database partitions.

Used with the SNAP_GET_Tbsp_V91, SNAP_GET_Tbsp_PART_V91, SNAP_GET_Tbsp_QUIESCER and SNAP_GET_CONTAINER_V91 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 108 on page 458 for a complete list of information that can be returned.

Syntax:

```

>> SNAP_GET_Tbsp_RANGE ( ( 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 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 database partitions.

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

SNAPTbsp_Range and Snap_Get_Tbsp_Range

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 partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the Snap_Get_Tbsp_Range table function.

Examples:

Select information on the table space range for the table space with `tblsp_id = 2` on the currently connected 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(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 108. 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 monitor element |
| Tbsp_Name | VARCHAR(128) | tablespace_name - Table Space Name monitor element |

SNAPTbsp_RANGE and SNAP_GET_Tbsp_RANGE

Table 108. 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_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_containers - Number of Containers in Range monitor element |
| RANGE_CONTAINER_ID | BIGINT | range_container_id - Range Container monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “GET SNAPSHOT command” in *Command Reference*
- “SNAPTbsp administrative view and SNAP_GET_Tbsp_V91 table function – Retrieve tablespace logical data group snapshot information” on page 441
- “SNAPTbsp_PART administrative view and SNAP_GET_Tbsp_PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information” on page 447
- “SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information” on page 351
- “SNAPTbsp_QUIESCER administrative view and SNAP_GET_Tbsp_QUIESCER table function – Retrieve quiescer table space snapshot information” on page 452
- “Database system monitor elements” in *System Monitor Guide and Reference*

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 on utilities from the utility_info logical data group.

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 109 on page 462 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPUTIL administrative view and EXECUTE privilege on the SNAP_GET_UTIL table function.

Example:

Retrieve a list of utilities and their states on all database partitions 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 partition, aggregate of all database partitions or all database partitions.

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 109 on page 462 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_UTIL ( [ dbpartitionnum ] ) ▶▶
```

The schema is SYSPROC.

Table function parameter:

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 database partitions. If this input option is not used, data will be returned from all database partitions.

If *dbpartitionnum* 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 partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_UTIL table function.

Example:

Retrieve a list of utility ids with their type and state for the currently connected database partition on database SAMPLE.

```
SELECT UTILITY_ID, UTILITY_TYPE, 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.

Information returned

SNAPUTIL and SNAP_GET_UTIL

Table 109. 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 monitor element. Unique to a database partition. |
| UTILITY_TYPE | VARCHAR(26) | utility_type - Utility Type monitor element. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • ASYNC_INDEX_CLEANUP • BACKUP • CRASH_RECOVERY • LOAD • REBALANCE • REDISTRIBUTE • REORG • RESTART_RECREATE_INDEX • RESTORE • ROLLFORWARD_RECOVERY • RUNSTATS |
| UTILITY_PRIORITY | INTEGER | utility_priority - Utility Priority monitor element. Priority if utility supports throttling, otherwise null. |
| UTILITY_DESCRIPTION | VARCHAR(2048) | utility_description - Utility Description monitor element. Can be null. |
| UTILITY_DBNAME | VARCHAR(128) | utility_dbname - Database Operated on by Utility monitor element |
| UTILITY_START_TIME | TIMESTAMP | utility_start_time - Utility Start Time monitor element |
| UTILITY_STATE | VARCHAR(10) | utility_state - Utility State monitor element. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • ERROR • EXECUTE • WAIT |
| UTILITY_INVOKER_TYPE | VARCHAR(10) | utility_invoker_type - Utility Invoker Type monitor element. 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 | The database partition from which the data was retrieved for this row. |

Table 109. Information returned by the SNAPUTIL administrative view and the SNAP_GET_UTIL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-------------|---|
| PROGRESS_LIST_ATTR | VARCHAR(10) | progress_list_attr - Current Progress List Attributes monitor element |
| PROGRESS_LIST_CUR_SEQ_NUM | INTEGER | progress_list_cur_seq_num - Current Progress List Sequence Number monitor element |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “LIST UTILITIES command” in *Command Reference*
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “Database system monitor elements” in *System Monitor Guide and Reference*
- “SNAPUTIL_PROGRESS administrative view and SNAP_GET_UTIL_PROGRESS table function – Retrieve progress logical data group snapshot information” on page 464

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.

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 110 on page 465 for a complete list of information that can be returned.

Authorization:

- SYSMON authority
- SELECT or CONTROL privilege on the SNAPUTIL_PROGRESS administrative view and EXECUTE privilege on the SNAP_GET_UTIL_PROGRESS table function.

Example:

Retrieve details on total and completed units of progress by utility ID.

```
SELECT 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 partition, aggregate of all database partitions or all database partitions.

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 110 on page 465 for a complete list of information that can be returned.

Syntax:

```
▶▶ SNAP_GET_UTIL_PROGRESS ( dbpartitionnum )
```

The schema is SYSPROC.

SNAPUTIL_PROGRESS and SNAP_GET_UTIL_PROGRESS

Table function parameter:

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 database partitions. If this input option is not used, data will be returned from all database partitions.

If *dbpartitionnum* 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 partition number.

Authorization:

- SYSMON authority
- EXECUTE privilege on the SNAP_GET_UTIL_PROGRESS table function.

Example:

Retrieve details on the progress of utilities on the currently connect partition.

```
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 110. 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 monitor element. Unique to a database partition. |
| PROGRESS_SEQ_NUM | INTEGER | progress_seq_num - Progress Sequence Number monitor element. If serial, the number of the phase. If concurrent, then could be NULL. |
| UTILITY_STATE | VARCHAR(16) | utility_state - Utility State monitor element. This interface returns a text identifier based on the defines in sqlmon.h and is one of: <ul style="list-style-type: none"> • ERROR • EXECUTE • WAIT |

SNAPUTIL_PROGRESS and SNAP_GET_UTIL_PROGRESS

Table 110. 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_DESCRIPTION | VARCHAR(2048) | progress_description - Progress Description monitor element |
| PROGRESS_START_TIME | TIMESTAMP | progress_start_time - Progress Start Time monitor element. Start time if the phase has started, otherwise NULL. |
| PROGRESS_WORK_METRIC | VARCHAR(16) | progress_work_metric - Progress Work Metric monitor element. 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 monitor element |
| PROGRESS_COMPLETED_UNITS | BIGINT | progress_completed_units - Completed Progress Work Units monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “LIST UTILITIES command” in *Command Reference*
- “SNAP_WRITE_FILE procedure” on page 313
- “Administrative views versus table functions” on page 3
- “SNAPUTIL_PROGRESS administrative view and SNAP_GET_UTIL_PROGRESS table function – Retrieve progress logical data group snapshot information” on page 464
- “Database system monitor elements” in *System Monitor Guide and Reference*

TBSP_UTILIZATION administrative view – Retrieve table space configuration and utilization information

The TBSP_UTILIZATION administrative view returns table space configuration and utilization information. The view is an SQL interface for the **LIST TABLESPACES** CLP command. Its information is based on the SNAPTbsp, SNAPTbsp_PART administrative views and TABLESPACES catalog view.

The schema is SYSIBMADM.

Authorization:

- SELECT or CONTROL privilege on the TBSP_UTILIZATION, SNAPTbsp, SNAPTbsp_PART administrative views and the SYSCAT.TABLESPACES catalog view.
- SYSMON, SYSCTRL, SYSMaint, or SYSADM authority is also required to access snapshot monitor data.

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      SMS          ...
1 TEMPSPACE1      SMS          ...
2 USERSPACE1      SMS          ...
3 SYSTOOLSPACE    SMS          ...
4 SYSTOOLSTMPSPACE SMS          ...
```

Output for this query (continued).

```
... TBSP_CONTENT_TYPE TBSP_STATE
... -----
... ANY                NORMAL
... SYSTEMP           NORMAL
... ANY                NORMAL
... ANY                NORMAL
... USRTEMP           NORMAL
```

Information returned:

Table 111. 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 monitor element |
| TBSP_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |

TBSP_UTILIZATION

Table 111. Information returned by the TBSP_UTILIZATION administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------|-------------|--|
| TBSP_TYPE | VARCHAR(10) | tablespace_type - Table Space Type monitor element. This interface returns a text identifier based on the 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 monitor element . This interface returns a text identifier based on the defines in sqlutil.h and is one of: <ul style="list-style-type: none">• ANY• LONG• SYSTEMP• USRTEMP |
| TBSP_CREATE_TIME | TIMESTAMP | Creation time of the table space. |

Table 111. Information returned by the TBSP_UTILIZATION administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|--------------|--|
| TBSP_STATE | VARCHAR(256) | <p>tablespace_state - Table Space State monitor element. 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_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

Table 111. Information returned by the TBSP_UTILIZATION administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|--------------|---|
| TBSP_UTILIZATION_PERCENT | BIGINT | The utilization of the table space as a percentage. Calculated as (used_pages/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 monitor element |
| TBSP_USABLE_PAGES | BIGINT | tablespace_usable_pages - Usable Pages in Table Space monitor element |
| TBSP_USED_PAGES | BIGINT | tablespace_used_pages - Used Pages in Table Space monitor element |
| TBSP_FREE_PAGES | BIGINT | tablespace_free_pages - Free Pages in Table Space monitor element |
| TBSP_PAGE_TOP | BIGINT | tablespace_page_top - Table Space High Water Mark monitor element |
| TBSP_PAGE_SIZE | INTEGER | tablespace_page_size - Table Space Page Size monitor element |
| TBSP_EXTENT_SIZE | INTEGER | tablespace_extent_size - Table Space Extent Size monitor element |
| TBSP_PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table Space Prefetch Size monitor element |
| TBSP_MAX_SIZE | BIGINT | tablespace_max_size - Maximum table space size monitor element |
| TBSP_INCREASE_SIZE | BIGINT | tablespace_increase_size - Increase size in bytes monitor element |
| TBSP_INCREASE_SIZE_PERCENT | SMALLINT | tablespace_increase_size_percent - Increase size by percent monitor element |
| TBSP_LAST_RESIZE_TIME | TIMESTAMP | tablespace_last_resize_time - Time of last successful resize monitor element |
| TBSP_LAST_RESIZE_FAILED | SMALLINT | tablespace_last_resize_failed - Last resize attempt failed monitor element |
| TBSP_USING_AUTO_STORAGE | SMALLINT | tablespace_using_auto_storage - Using automatic storage monitor element |
| TBSP_AUTO_RESIZE_ENABLED | SMALLINT | tablespace_auto_resize_enabled - Auto-resize enabled monitor element |
| 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 monitor element |

Table 111. Information returned by the TBSP_UTILIZATION administrative view (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------|--------------|--|
| REMARKS | VARCHAR(254) | User-provided comment. |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SYSCAT.TABLESPACES catalog view” in *SQL Reference, Volume 1*
- “Database system monitor elements” in *System Monitor Guide and Reference*
- “Authorization for administrative views” on page 6
- “LIST TABLESPACES command” in *Command Reference*
- “SNAPTbsp administrative view and SNAP_GET_TBSP_V91 table function – Retrieve tablespace logical data group snapshot information” on page 441
- “SNAPTbsp_PART administrative view and SNAP_GET_TBSP_PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information” on page 447

TOP_DYNAMIC_SQL administrative view – Retrieve information on 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. These are the queries that should get focus to ensure they are well tuned.

The schema is SYSIBMADM.

Authorization:

- SELECT or CONTROL privilege on the TOP_DYNAMIC_SQL and SNAPDYN_SQL administrative views.
- SYSMON, SYSCTRL, SYSMOINT, or SYSADM authority is also required to access snapshot monitor data.

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
```

Information returned:*Table 112. 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_compilations - Statement Compilations monitor element |
| AVERAGE_EXECUTION_TIME_S | BIGINT | Average execution time. |
| STMT_SORTS | BIGINT | stmt_sorts - Statement Sorts monitor element |
| SORTS_PER_EXECUTION | BIGINT | Number of sorts per statement execution. |
| STMT_TEXT | CLOB(2 M) | stmt_text - SQL Dynamic Statement Text monitor element |
| DBPARTITIONNUM | SMALLINT | The database partition from which the data was retrieved for this row. |

Related tasks:

- “Capturing database system snapshots using snapshot administrative views and table functions” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information” on page 387
- “Authorization for administrative views” on page 6
- “Database system monitor elements” in *System Monitor Guide and Reference*

SQL procedure administrative SQL routines and views

GET_ROUTINE_OPTS

▶▶—GET_ROUTINE_OPTS—(—)—▶▶

The schema is SYSPROC.

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.

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
```

Related reference:

- “SET_ROUTINE_OPTS ” on page 479
- “Supported administrative SQL routines and views” on page 8

GET_ROUTINE_SAR

```

▶▶—GET_ROUTINE_SAR—————▶▶
▶—(—sarblob—,—type—,—routine-name-string—————)————▶▶
     [,—hide-body-flag—]

```

The schema is SYSFUN.

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. The information is retrieved into a single BLOB string representing an SQL archive file. The invoker of the GET_ROUTINE_SAR procedure must have DBADM authority.

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 desired 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).

Related reference:

- "Supported administrative SQL routines and views" on page 8

PUT_ROUTINE_SAR

```

▶▶ PUT_ROUTINE_SAR ( ( --sarblob [ , --new-owner , --use-register-flag ] ) ) ▶▶

```

The schema is SYSFUN.

The PUT_ROUTINE_SAR procedure passes the necessary file to create an SQL routine at the server and then defines the routine. The invoker of the PUT_ROUTINE_SAR procedure must have DBADM authority.

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" and "42704" are the SQLCODE and SQLSTATE, respectively, 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.

Related reference:

- "Supported administrative SQL routines and views" on page 8

REBIND_ROUTINE_PACKAGE

►►—REBIND_ROUTINE_PACKAGE—(—*type*—,—*routine-name-string*—,—*resolve*—)————►◄

The schema is SYSPROC.

The REBIND_ROUTINE_PACKAGE procedure rebinds the package associated with an SQL procedure. It 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.

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 value of the CURRENT_SCHEMA special register when the routine is processed. The *routine-name-string* cannot include double quotation marks (").

resolve

An input argument of type VARCHAR(12) that specifies which binding semantics should be used. A value of 'ANY' indicates that any of the functions and types in the SQL path are considered for function and type resolution. A value of 'CONSERVATIVE' indicates that only functions and types in the SQL path that were defined before the last explicit bind time stamp are considered for function and type resolution.

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 desired routine must be used.

Related reference:

- "Supported administrative SQL routines and views" on page 8

SET_ROUTINE_OPTS

►►—SET_ROUTINE_OPTS—(*—character-expression—*)—►►

The schema is SYSPROC.

The SET_ROUTINE_OPTS procedure sets the options that are to be used for the creation of SQL procedures in the current session. This setting overrides the instance-wide setting specified in the DB2_SQLROUTINE_PREPOPTS registry variable.

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:

```
CALL SYSPROC.SET_ROUTINE_OPTS(CAST (NULL AS VARCHAR(1)))
```

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “GET_ROUTINE_OPTS ” on page 474
- “Query compiler variables” in *Performance Guide*

Stepwise redistribute administrative SQL routines

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

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 respective 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 respective 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.

ANALYZE_LOG_SPACE

Table 113. 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 desired 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. |

Related concepts:

- “Data redistribution” in *Performance Guide*
- “Partitioned database environments” in *Administration Guide: Planning*
- “Distribution keys” in *Administration Guide: Planning*

Related tasks:

- “Defining distribution keys” in *Administration Guide: Implementation*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Redistributing data using step-wise redistribute procedures” in *Performance Guide*
- “GENERATE_DISTFILE procedure – Generate a data distribution file” on page 483
- “GET_SWRD_SETTINGS procedure – Retrieve redistribute information” on page 485
- “SET_SWRD_SETTINGS procedure – Create or change redistribute registry” on page 488
- “STEPWISE_REDISTRIBUTE_DBPG procedure – Redistribute part of database partition group” on page 491

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.

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.

Related concepts:

- “Data redistribution” in *Performance Guide*
- “Partitioned database environments” in *Administration Guide: Planning*
- “Distribution keys” in *Administration Guide: Planning*

Related tasks:

- “Defining distribution keys” in *Administration Guide: Implementation*

GENERATE_DISTFILE

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Redistributing data using step-wise redistribute procedures” in *Performance Guide*
- “ANALYZE_LOG_SPACE procedure – Retrieve log space analysis information” on page 480
- “GET_SWRD_SETTINGS procedure – Retrieve redistribute information” on page 485
- “SET_SWRD_SETTINGS procedure – Create or change redistribute registry” on page 488
- “STEPWISE_REDISTRIBUTE_DBPG procedure – Redistribute part of database partition group” on page 491

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 114, 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 rest of the output parameters can be null.

Table 114. Bitwise field indentifiers

| 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.

GET_SWRD_SETTINGS

- 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:

EXECUTE privilege on the GET_SWRD_SETTINGS procedure.

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.

Related concepts:

- “Data redistribution” in *Performance Guide*
- “Partitioned database environments” in *Administration Guide: Planning*
- “Distribution keys” in *Administration Guide: Planning*
- “Distribution maps” in *Administration Guide: Planning*

Related tasks:

- “Defining distribution keys” in *Administration Guide: Implementation*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Redistributing data using step-wise redistribute procedures” in *Performance Guide*
- “ANALYZE_LOG_SPACE procedure – Retrieve log space analysis information” on page 480
- “GENERATE_DISTFILE procedure – Generate a data distribution file” on page 483
- “SET_SWRD_SETTINGS procedure – Create or change redistribute registry” on page 488
- “STEPWISE_REDISTRIBUTE_DBPG procedure – Redistribute part of database partition group” on page 491

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 115 indicating the target fields to be written or overwritten into the redistribute settings registry.

Table 115. 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:

EXECUTE privilege on the SET_SWRD_SETTINGS procedure.

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.

SET_SWRD_SETTINGS

```
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.

Related concepts:

- “Data redistribution” in *Performance Guide*
- “Partitioned database environments” in *Administration Guide: Planning*
- “Distribution keys” in *Administration Guide: Planning*
- “Distribution maps” in *Administration Guide: Planning*

Related tasks:

- “Defining distribution keys” in *Administration Guide: Implementation*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Redistributing data using step-wise redistribute procedures” in *Performance Guide*
- “ANALYZE_LOG_SPACE procedure – Retrieve log space analysis information” on page 480
- “GENERATE_DISTFILE procedure – Generate a data distribution file” on page 483
- “GET_SWRD_SETTINGS procedure – Retrieve redistribute information” on page 485
- “STEPWISE_REDISTRIBUTE_DBPG procedure – Redistribute part of database partition group” on page 491

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.

Authorization:

- EXECUTE privilege on the STEPWISE_REDISTRIBUTE_DBPG procedure
- SYSADM, SYSCTRL or DBADM

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)
```

STEPWISE_REDISTIBUTE_DBPG

For a full usage example of the stepwise redistribute procedures, refer to [Redistributing data using step-wise redistribute procedures](#)

Usage notes:

If the registry value for *processState* is updated to 1 using the SET_SWRD_SETTINGS procedure after the STEPWISE_REDISTIBUTE_DBPG procedure execution is started, the process stops at the beginning to the next step and a warning message is returned.

Since SQL COMMIT statement is called by the redistribute process, running the redistribute process under a Type-2 connection is not supported.

Related concepts:

- “Data redistribution” in *Performance Guide*
- “Partitioned database environments” in *Administration Guide: Planning*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Redistributing data using step-wise redistribute procedures” in *Performance Guide*
- “ANALYZE_LOG_SPACE procedure – Retrieve log space analysis information” on page 480
- “GENERATE_DISTFILE procedure – Generate a data distribution file” on page 483
- “GET_SWRD_SETTINGS procedure – Retrieve redistribute information” on page 485
- “SET_SWRD_SETTINGS procedure – Create or change redistribute registry” on page 488

Storage management tool administrative SQL routines

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 116. 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 |
| 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 |

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.

CAPTURE_STORAGEMGMT_INFO

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.

Authorization:

- EXECUTE privilege on the CAPTURE_STORAGEMGMT_INFO procedure.
- EXECUTE privilege on the SYSPROC.DB_PARTITIONS, SYSPROC.SNAP_GET_CONTAINER, SYSPROC.SNAPSHOT_CNTRFS table functions.
- SELECT privilege on SYSCAT.TABLES, SYSCAT.TABLESPACES, SYSCAT.NODEGROUPDEF, SYSCAT.DATABASEPARTITIONS, SYSCAT.DATAPARTITIONEXPRESSION, SYSCAT.INDEXES, and SYSCAT.COLUMNS.

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “CREATE_STORAGEMGMT_TABLES procedure – Create storage management tables” on page 495
- “DROP_STORAGEMGMT_TABLES procedure – Drop all storage management tables” on page 497
- “Storage management view” in *Administration Guide: Planning*
- “Storage management view tables” in *Administration Guide: Planning*

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 either:

- 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.
- SYSADM or DBADM authority

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

Related reference:

CREATE_STORAGEMGMT_TABLES

- “Supported administrative SQL routines and views” on page 8
- “CAPTURE_STORAGEMGMT_INFO procedure – Retrieve storage-related information for a given root object” on page 493
- “DROP_STORAGEMGMT_TABLES procedure – Drop all storage management tables” on page 497
- “Storage management view” in *Administration Guide: Planning*

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:

- SYSADM or DBADM authority
- DROPIN privilege on the schema for these tables
- CONTROL privilege on these tables

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “CAPTURE_STORAGEMGMT_INFO procedure – Retrieve storage-related information for a given root object” on page 493
- “CREATE_STORAGEMGMT_TABLES procedure – Create storage management tables” on page 495
- “Storage management view” in *Administration Guide: Planning*

Miscellaneous administrative SQL routines and views

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.

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 prior to 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 on 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 117. ADMIN_COPY_SCHEMA errortab format

| Column name | Data type | Description |
|-----------------|--------------|--|
| OBJECT_SCHEMA | VARCHAR(128) | Schema name of the object for which the copy command failed. |
| OBJECT_NAME | VARCHAR(128) | Name of the object for which the copy command failed. |
| OBJECT_TYPE | VARCHAR(30) | Type of object. |
| SQLCODE | INTEGER | The error SQLCODE. |
| SQLSTATE | CHAR(5) | The error SQLSTATE. |
| ERROR_TIMESTAMP | TIMESTAMP | Time of failure for the operation that failed. |

ADMIN_COPY_SCHEMA

Table 117. ADMIN_COPY_SCHEMA errortab format (continued)

| Column name | Data type | Description |
|-------------|-----------|---|
| 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. |

Authorization:

In order for the schema copy to be successful, the user ID calling this procedure must have the appropriate object creation authorities including both the authority to select from the source tables, and the authority to perform a load. 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.

Example:

```
CALL SYSPROC.ADMIN_COPY_SCHEMA('SOURCE_SCHEMA', 'TARGET_SCHEMA',  
    'COPY', NULL, 'SOURCETS1', SOURCETS2', 'TARGETTS1, TARGETTS2,  
    SYS_ANY', 'ERRORSCHEMA', 'ERRORNAME')
```

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 on how to resolve this problem, see “Copying a schema”.

Usage notes:

- Qualified objects within the objects being copied are not modified. The ADMIN_COPY_SCHEMA procedure only changes the qualifying schema of the object being created, not any data within those objects.
- This procedure does not support copying the following objects:
 - index extensions
 - nicknames
 - packages
 - typed tables
 - user-defined structured types (and their transform functions)
 - typed views

- jars (Java™ routine archives)
- staging tables
- If one of the above 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 recreated 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 partitioning key selection is preferred, the partitioning key should be defined on the tables before the copy schema operation is undertaken. Altering the partitioning 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 rollback 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.

ADMIN_COPY_SCHEMA

Related concepts:

- “SYSTOOLSPACE and SYSTOOLSTMPSPACE table spaces” in *Administration Guide: Planning*
- “Table locking, table states and table space states” in *Data Movement Utilities Guide and Reference*

Related tasks:

- “Copying a schema” in *Administration Guide: Implementation*
- “Restarting a failed copy schema operation” in *Administration Guide: Implementation*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “ADMIN_DROP_SCHEMA procedure – Drop a specific schema and its objects” on page 503
- “CREATE TABLE statement” in *SQL Reference, Volume 2*

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 118. ADMIN_DROP_SCHEMA errortab format

| Column name | Data type | Description |
|-----------------|--------------|--|
| OBJECT_SCHEMA | VARCHAR(128) | Schema name of the object for which the drop command failed. |
| OBJECT_NAME | VARCHAR(128) | Name of the object for which the drop command failed. |
| OBJECT_TYPE | VARCHAR(30) | Type of object. |
| SQLCODE | INTEGER | The error SQLCODE. |
| SQLSTATE | CHAR(5) | The error SQLSTATE. |
| ERROR_TIMESTAMP | TIMESTAMP | Time that the drop command failed. |

ADMIN_DROP_SCHEMA

Table 118. ADMIN_DROP_SCHEMA errortab format (continued)

| Column name | Data type | Description |
|-------------|-----------|---|
| STATEMENT | CLOB(2 M) | DDL for the failing object. |
| DIAGTEXT | CLOB(2 K) | Error message text for the failed drop command. |

Authorization:

Drop authority is needed on all objects being removed for the user calling this procedure.

EXECUTE privilege on the ADMIN_DROP_SCHEMA procedure is also needed.

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
 - user-defined structured types (and their transform functions)
 - typed views
 - jars (Java routine archives)
 - staging tables
- If one of the above 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.

Related concepts:

- “SYSTOOLSPACE and SYSTOOLSTMPSPACE table spaces” in *Administration Guide: Planning*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “ADMIN_COPY_SCHEMA procedure – Copy a specific schema and its objects” on page 498
- “DROP statement” in *SQL Reference, Volume 2*

ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function – Retrieve size and state information for tables

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.

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.

The schema is SYSIBMADM.

Refer to the “ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function metadata” on page 508 table for a complete list of information that can be returned.

Authorization:

SELECT or CONTROL privilege on the ADMINTABINFO administrative view and EXECUTE privilege on the “ADMIN_GET_TAB_INFO table function.”

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: Identify which tables are using type-1 indexes and require a reorganization to convert to type-2 indexes.

```
SELECT TABSCHEMA, TABNAME FROM SYSIBMADM.ADMINTABINFO
WHERE INDEX_TYPE = 1
```

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 “ADMINTABINFO administrative view and ADMIN_GET_TAB_INFO table function metadata” on page 508 table for a complete list of information that can be returned.

Syntax:

```
►► ADMIN_GET_TAB_INFO(—tabschema—, —tablename—)◄◄
```

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.

Authorization:

EXECUTE privilege on the ADMIN_GET_TAB_INFO table function.

Example:

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) 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) as total_l_size
FROM TABLE( SYSPROC.ADMIN_GET_TAB_INFO( 'DBUSER1', 'EMPLOYEE' )) AS T
```

Usage notes:

- If both the *tabschema* and *tablename* are specified, information is returned for that specific table only.
- If the *tabschema* is specified but *tablename* is empty (") or NULL, information is returned for all tables in the given schema.
- If the *tabschema* is empty (") or NULL and *tablename* is specified, the value of CURRENT_SCHEMA is assumed and information is returned only for the specified table.
- If both *tabschema* and *tablename* are empty (") or NULL, information is returned for all tables.
- If *tabschema* or *tablename* do not exist, or *tablename* 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

ADMINTABINFO and ADMIN_GET_TAB_INFO

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.
- The logical size reported for LOB objects created before DB2 UDB Version 8 might be larger than the physical size if the objects have not yet been reorganized.

ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO table function metadata

Table 119. ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO table function metadata

| Column name | Data type | Description |
|-------------------|--------------|--|
| TABSHEMA | VARCHAR(128) | Schema name. |
| TABNAME | VARCHAR(128) | Table name. |
| TABTYPE | CHAR(1) | Table type: <ul style="list-style-type: none"> • 'H' = hierarchy table • 'S' = materialized query table • 'T' = table |
| DBPARTITIONNUM | SMALLINT | Database partition number. |
| DATA_PARTITION_ID | INTEGER | Data partition number. |
| 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. <p>Note: Rollforward through an unrecoverable load will put a table into the unavailable state.</p> |

Table 119. ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO table function metadata (continued)

| Column name | Data type | Description |
|--------------------|-----------|--|
| 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. |

ADMINTABINFO and ADMIN_GET_TAB_INFO

Table 119. ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO table function metadata (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. |

Table 119. ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO table function metadata (continued)

| Column name | Data type | Description |
|--------------------|-----------|---|
| 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. |

ADMINTABINFO and ADMIN_GET_TAB_INFO

Table 119. ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO table function metadata (continued)

| Column name | Data type | Description |
|-------------------|-----------|--|
| 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. |

ADMINTABINFO and ADMIN_GET_TAB_INFO

Table 119. ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO table function metadata (continued)

| Column name | Data type | Description |
|----------------------|-------------|---|
| INPLACE_REORG_STATUS | VARCHAR(10) | Current status of an inplace table reorganization on the table. The status value can be one of the following: <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: <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. |

ADMINTABINFO and ADMIN_GET_TAB_INFO

Table 119. ADMINTABINFO administrative view and the ADMIN_GET_TAB_INFO table function metadata (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. If no indexes are defined on the table, 'N' will also be returned, since there are no indexes that require a rebuild. |
| 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. |

Related concepts:

- "Index structure" in *Performance Guide*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SYSCAT.TABLES catalog view” in *SQL Reference, Volume 1*
- “Administrative views versus table functions” on page 3

ALTOBJ

The ALTOBJ procedure parses an input CREATE TABLE statement serving as the target data definition language (DDL) for an existing table that is to be altered. 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 DB2 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.

'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:

EXECUTE privilege on the ALTOBJ procedure.

DBADM with LOAD authority, and SETSESSIONUSER are also required.

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, ?)
```

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:

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.

ALTOBJ

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.

Related concepts:

- "SYSTOOLSPACE and SYSTOOLSTMPSPACE table spaces" in *Administration Guide: Planning*

Related reference:

- "Supported administrative SQL routines and views" on page 8
- "CREATE TABLE statement" in *SQL Reference, Volume 2*
- "GRANT (SETSESSIONUSER Privilege) statement" in *SQL Reference, Volume 2*

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.

Syntax:

►► APPLICATION_ID (—) ◀◀

The schema is SYSFUN.

Example:

```
SELECT APPLICATION_ID() AS APPL_ID FROM SYSIBM.SYSDUMMY1
```

Related reference:

- “appl_id - Application ID monitor element” in *System Monitor Guide and Reference*
- “Supported administrative SQL routines and views” on page 8

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(2 M) that contains a compilation environment provided by a deadlock event monitor.

The function returns a table of two columns (see Table 120): NAME VARCHAR(256) and VALUE VARCHAR(1024). The possible values for the compilation environment element names are described in Table 121 on page 521.

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 DB2 database manager. If the compilation environment is from a different DB2 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
```

Example 2: Request a specific element (the default schema) of a compilation environment.

```
SELECT NAME, VALUE
FROM TABLE(SYSPROC.COMPILATION_ENV(:hv1)) AS t
WHERE NAME = 'SCHEMA'
```

Information returned:

Table 120. Information returned by the COMPILATION_ENV table function

| Column name | Data type | Description |
|-------------|---------------|---|
| NAME | VARCHAR(256) | Element of compilation environment. See Table 121 on page 521 for more details. |
| VALUE | VARCHAR(1024) | Value of the element. |

Table 121. 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. |
| 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. |
| MIN_DEC_DIV_3 | The requested decimal computational scale passed to the SQL compiler. The value is obtained from the <i>min_dec_div_3</i> database configuration parameter. |
| 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. |

Related reference:

COMPILATION_ENV

- “Supported administrative SQL routines and views” on page 8
- “CREATE EVENT MONITOR statement” in *SQL Reference, Volume 2*
- “SET COMPILATION ENVIRONMENT statement” in *SQL Reference, Volume 2*

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:

SELECT or CONTROL privilege on the CONTACTGROUPS administrative view and EXECUTE privilege on the ADMIN_GET_CONTACTGROUPS table function.

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 122. 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. |

Related tasks:

- “Notification and contact list setup and configuration” in *Administration Guide: Implementation*

CONTACTGROUPS

Related reference:

- “contact_host - Location of contact list configuration parameter” in *Performance Guide*
- “Supported administrative SQL routines and views” on page 8
- “Administrative views versus table functions” on page 3
- “Authorization for administrative views” on page 6

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:

SELECT or CONTROL privilege on the CONTACTS administrative view and EXECUTE privilege on the ADMIN_GET_CONTACTS table function.

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 123. Information returned by the CONTACTS administrative view

| Column name | Data type | Description |
|-----------------|--------------|---|
| NAME | VARCHAR(128) | Name of contact. |
| TYPE | VARCHAR(5) | Type of contact: • 'EMAIL' • 'PAGE' |
| ADDRESS | VARCHAR(128) | SMTP mailbox address of the recipient. For example, joe@somewhere.org. |
| 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. |

Related tasks:

- “Notification and contact list setup and configuration” in *Administration Guide: Implementation*

Related reference:

CONTACTS

- “Supported administrative SQL routines and views” on page 8
- “contact_host - Location of contact list configuration parameter” in *Performance Guide*
- “Administrative views versus table functions” on page 3
- “Authorization for administrative views” on page 6

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:

SELECT or CONTROL privilege on the DB_HISTORY administrative view and EXECUTE privilege on the ADMIN_LIST_HIST table function.

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 124. Information returned by the DB_HISTORY administrative view

| Column name | Data type | Description |
|----------------|--------------|--|
| DBPARTITIONNUM | SMALLINT | Database partition number. |
| EID | BIGINT | Number that uniquely identifies an entry in the history file. |
| START_TIME | VARCHAR(14) | Timestamp marking the start of a logged event. |
| SEQNUM | SMALLINT | Sequence number. |
| END_TIME | VARCHAR(14) | Timestamp marking the end of a logged event. |
| FIRSTLOG | VARCHAR(254) | Name of the earliest transaction log associated with an event. |
| LASTLOG | VARCHAR(254) | Name of the latest transaction log associated with an event. |
| BACKUP_ID | VARCHAR(24) | Backup identifier or unique table identifier. |
| TABSCHEMA | VARCHAR(128) | Table schema. |
| TABNAME | VARCHAR(128) | Table name. |
| COMMENT | VARCHAR(254) | System-generated comment text associated with a logged event. |

Table 124. Information returned by the DB_HISTORY administrative view (continued)

| Column name | Data type | Description |
|---------------|--------------|--|
| CMD_TEXT | CLOB(2 M) | Data definition language associated with a logged event. |
| NUM_TBSPS | INTEGER | Number of table spaces associated with a logged event. |
| TBSPNAMES | CLOB(5 M) | Names of the table spaces associated with a logged event. |
| OPERATION | CHAR(1) | Operation identifier. See Table 125 on page 529 for possible values. |
| OPERATIONTYPE | CHAR(1) | Action identifier for an operation. See Table 125 on page 529 for possible values. |
| OBJECTTYPE | CHAR(1) | Identifier for the target object of an operation. The possible values are: D for full database, P for table space, and T for table. |
| LOCATION | VARCHAR(255) | Full path name for files, such as backup images or load input file, that are associated with logged events. |
| DEVICETYPE | CHAR(1) | Identifier for the device type associated with a logged event. This field determines how the LOCATION field is interpreted. The possible values are: A for TSM, C for client, D for disk, K for diskette, L for local, O for other (for other vendor device support), P for pipe, Q for cursor, S for server, T for tape, and U for user exit. |
| ENTRY_STATUS | CHAR(1) | Identifier for the status of an entry in the history file. The possible values are: 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. |

Table 124. Information returned by the DB_HISTORY administrative view (continued)

| Column name | Data type | Description |
|-------------|-------------|---|
| 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 digits 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). |
| SQLERRD2 | INTEGER | See SQLCA (SQL communications area). |
| SQLERRD3 | INTEGER | See SQLCA (SQL communications area). |
| SQLERRD4 | INTEGER | See SQLCA (SQL communications area). |
| SQLERRD5 | INTEGER | See SQLCA (SQL communications area). |
| SQLERRD6 | INTEGER | See SQLCA (SQL communications area). |
| SQLWARN | VARCHAR(11) | A set of warning indicators, each containing a blank or 'W'. See SQLCA (SQL communications area). |
| 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 125. OPERATION and OPERATIONTYPE values

| Operation value | Operation value description | Operation type |
|-----------------|-----------------------------|----------------|
| A | Add table space | None |

Table 125. OPERATION and OPERATIONTYPE values (continued)

| Operation value | Operation value description | Operation type |
|-----------------|-----------------------------|--|
| 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 125. 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 |

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “LIST HISTORY command” in *Command Reference*
- “SQLCA (SQL communications area)” in *SQL Reference, Volume 1*
- “Administrative views versus table functions” on page 3
- “Authorization for administrative views” on page 6

DBPATHS administrative view – Retrieve database paths

The DBPATHS administrative view returns the values for database paths required for tasks such as split mirror backups.

The schema is SYSIBMADM.

Authorization:

SELECT or CONTROL privilege on the DBPATHS administrative view and EXECUTE privilege on ADMIN_LIST_DB_PATHS table function.

Example:

Retrieve all database paths.

```
SELECT * FROM SYSIBMADM.DBPATHS
```

The following is an example of output for this query.

```
DBPARTITIONNUM TYPE ...
-----
0 LOGPATH ...
0 MIRRORLOGPATH ...
0 DB_STORAGE_PATH ...
0 DB_STORAGE_PATH ...
0 TBSP_CONTAINER ...
0 TBSP_CONTAINER ...
0 TBSP_CONTAINER ...
0 TBSP_DIRECTORY ...
0 TBSP_DIRECTORY ...
0 LOCAL_DB_DIRECTORY ...
0 DBPATH ...
```

11 record(s) selected.

Output for this query (continued).

```
... PATH
... -----
... S:\dbfiles\INST5\NODE0000\SQL00001\SQLGDIR\
... S:\mirrorlogs\NODE0000\
... S:\dbfiles\
... S:\dbfile2\
... S:\dbfiles\INST5\NODE0000\SQL00001\TS3
... S:\dbfiles\INST5\NODE0000\SQL00001\long3
... S:\dbfiles\INST5\NODE0000\SQL00001\regular05
... S:\dbfiles\INST5\NODE0000\SQL00001\usertemp3\
... S:\dbfiles\INST5\NODE0000\SQL00001\systemp3\
... S:\dbfiles\INST5\NODE0000\SQLDBDIR\
... S:\dbfiles\INST5\NODE0000\SQL00001\
```

Information returned:

Table 126. Information returned by the DBPATHS administrative view

| Column name | Data type | Description |
|----------------|-----------|----------------------------|
| DBPARTITIONNUM | SMALLINT | Database partition number. |

Table 126. Information returned by the DBPATHS administrative view (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 127 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 127. 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. |
| LOGPATH_DEVICE | Raw device for 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. |

Table 127. TYPE column values (continued)

| Type value | Description |
|------------|--|
| | <ul style="list-style-type: none"> For table spaces using automatic storage, both used and unused storage paths are returned. The unused automatic storage paths are needed in case the split mirror backup is restored. Consider the following example: A split mirror backup is taken on a production system. After the backup completes, 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 split mirror backup. At this point, it is necessary to roll forward the logs from the production database. In order to roll forward the logs, all of the automatic storage paths are required since all automatic storage paths are now in use. Table space containers 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 LOGPATH and MIRRORLOGPATH are the values stored in memory. Changed values stored on disk, which are only applicable after a database restart, are not returned. If output from <code>SELECT * FROM SYSIBMADM.DBPATHS</code> is being used to create a <code>db2relocatedb</code> configuration file (a file containing the configuration information necessary for relocating a database), the <code>DBPATH</code> output must be modified appropriately before it can be used in the configuration file. For example, the following <code>DBPATH</code> output: <code>/storage/svtdbm3/svtdbm3/NODE0000/SQL00001/</code> can be used to specify the <code>DB_PATH</code> parameter in a <code>db2relocatedb</code> configuration file as follows: <code>DB_PATH=/storage/svtdbm3,/storage_copy2/svtdbm3</code> |
| | <ul style="list-style-type: none"> The <code>LOCAL_DB_DIRECTORY</code> path might contain information belonging to multiple databases. Because the <code>sqlbdbir</code> is not separated for multiple databases created in the same directory, ensure that the target system to which files will be copied does not have any databases already existing 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 were not intended to be split. |

Restriction:

This administrative view cannot be called when the database is in `WRITE SUSPEND` mode. The database administrator must ensure that the physical layout of the database does not change in the time between the invocation of the view and the activation of the `WRITE SUSPEND` mode, which is needed to perform the split mirror operation. The split mirror backup image might not be restored successfully if, for example, the table space layout changed in that time.

Related concepts:

- “Automatic storage databases” in *Administration Guide: Implementation*
- “High availability through online split mirror and suspended I/O support” in *Data Recovery and High Availability Guide and Reference*

Related tasks:

- “Using a split mirror as a backup image” in *Data Recovery and High Availability Guide and Reference*
- “Using a split mirror as a standby database” in *Data Recovery and High Availability Guide and Reference*
- “Using a split mirror to clone a database” in *Data Recovery and High Availability Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Administrative views versus table functions” on page 3
- “Authorization for administrative views” on page 6
- “db2relocatedb - Relocate database command” in *Command Reference*
- “SET WRITE command” in *Command Reference*

EXPLAIN_GET_MSGS

```

▶▶—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.

The EXPLAIN_GET_MSGS table function queries the EXPLAIN_DIAGNOSTIC and EXPLAIN_DIAGNOSTIC_DATA Explain tables, and returns formatted messages.

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 DB2 server, the value is ignored.

The function returns a table as shown below.

Table 128. 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 DB2 server. |
| MSG | VARCHAR(4096) | Formatted message text. |

Example

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),
```

EXPLAIN_GET_MSGS

```
'en_US'))  
AS REGISTRYINFO  
WHERE EXPLAIN_TIME >= (CURRENT TIME - 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".
```

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “EXPLAIN_STATEMENT table” in *SQL Reference, Volume 1*

GET_DBSIZE_INFO

The GET_DBSIZE_INFO procedure calculates the database size and maximum capacity.

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: $dbsize = \text{sum}(\text{used_pages} * \text{page_size})$ 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: $dbcapacity = \text{SUM}(\text{DMS usable_pages} * \text{page size}) + \text{SUM}(\text{SMS container size} + \text{file system free size per container})$. 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.

Authorization:

- SYSMON authority
- EXECUTE privilege on the GET_DBSIZE_INFO procedure

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

GET_DBSIZE_INFO

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

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.

Related concepts:

- "SYSTOOLSPACE and SYSTOOLSTMPSPACE table spaces" in *Administration Guide: Planning*

Related reference:

- "Supported administrative SQL routines and views" on page 8

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:

SELECT or CONTROL privilege on the NOTIFICATIONLIST administrative view and EXECUTE privilege on the HEALTH_GET_NOTIFICATION_LIST table function.

Example:

Retrieve all contacts that will receive notification of health alerts.

```
SELECT * FROM SYSIBMADM.NOTIFICATIONLIST
```

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 129. 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' |

Related tasks:

- “Enabling health alert notification” in *System Monitor Guide and Reference*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Administrative views versus table functions” on page 3
- “Authorization for administrative views” on page 6

PDLOGMSGs_LAST24HOURS and PD_GET_LOG_MSGS

```

...
...          0 CAPTAIN          4239374 db2agent (CAPTAIN) 0 ...
...          0 CAPTAIN          4239374 db2agent (CAPTAIN) 0 ...
...
...
...
...          0 CAPTAIN          4239374 db2agent (CAPTAIN) 0 ...
...
...

```

Output from this query (continued).

```

...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 ...
...
...

```

Output from this query (continued).

```

... FUNCTION          PROBE  MSGNUM      MSGTYPE ...
-----
... sqloSleepInstance      38      504 ADM      ...
...
...
...
...
...
...
... sqlMarkDBad           10      7518 ADM     ...
... sqlrr_dump_ffdc       10      1 ADM       ...
...
...
...
... sqlrr_dump_ffdc       10      1 ADM       ...
...

```

Output from this query (continued).

```

... 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 130 on page 548 for a complete list of information that can be returned.

Syntax:

►► PD_GET_LOG_MSGS (—*oldest_timestamp*—) ◀◀

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.

Authorization:

EXECUTE privilege on the PD_GET_LOG_MSGS table function.

Examples:

Example 1: Retrieve all notification messages logged for database SAMPLE on instance DB2 in the last week for all database partitions. Report messages in chronological order.

```
SELECT TIMESTAMP, APPL_ID, DBPARTITIONNUM, MSG
FROM TABLE ( PD_GET_LOG_MSGS( CURRENT_TIMESTAMP - 7 DAYS)) AS T
WHERE INSTANCENAME = 'DB2' 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).

PDLOGMSGS_LAST24HOURS and PD_GET_LOG_MSGS

```
... 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 DB2 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 = 'DB2' 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 DB2 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.

PDLOGMSGG_LAST24HOURS and PD_GET_LOG_MSGG

| 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_MSGG(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. |
| ... | | |
| ... | | |
| ... | | |
| ... | | |
| ... | | |
| ... | | |
| ... | | |
| ... | | |
| ... | | |

Information returned

PDLOGMSGs_LAST24HOURS and PD_GET_LOG_MSGS

Note: In a partitioned database 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 130. 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 | The database partition that generated the message. For a non partitioned database environment, 0 is returned. |
| 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) | ID of the application for which the process is working. |
| COMPONENT | VARCHAR(255) | The name of the DB2 component that wrote the message. For messages written by user applications using the db2AdminMsgWrite API, "User Application" is returned. |
| FUNCTION | VARCHAR(255) | The name of the DB2 function that is providing the message. For messages written by user applications using the db2AdminMsgWrite API, "User Function" is returned. |
| PROBE | INTEGER | Unique internal identifier that allows DB2 Customer Support and Development to locate the point in the DB2 source code that generated the message. |
| MSGNUM | INTEGER | The numeric message number for the error or event. |

PDLOGMSGG_LAST24HOURS and PD_GET_LOG_MSGG

Table 130. Information returned by the PDLOGMSGG_LAST24HOURS administrative view and the PD_GET_LOG_MSGG table function (continued)

| Column name | Data type | Description |
|-------------|-------------|---|
| 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(16 KB) | Notification log message text. |

Related concepts:

- “Interpreting administration notification log file entries” in *Troubleshooting Guide*
- “Administration notification log” in *Administration Guide: Implementation*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “Administrative views versus table functions” on page 3

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', specifies a fully qualified table name, or accepts one of the following values: ALL, USER, or SYSTEM. If *scope* has a value of 'S', specifies a schema name.

Authorization:

- SELECT privilege on catalog tables.
- EXECUTE privilege on the REORGCHK_IX_STATS procedure.

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 131. Information returned by the REORGCHK_IX_STATS procedure

| Column name | Data type | Description |
|--------------|--------------|---|
| TABLE_SCHEMA | VARCHAR(128) | Schema name. |
| TABLE_NAME | VARCHAR(128) | Table name. |
| INDEX_SCHEMA | VARCHAR(128) | Index schema name. |
| INDEX_NAME | VARCHAR(128) | Index name. |
| 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. |

Table 131. Information returned by the REORGCHK_IX_STATS procedure (continued)

| Column name | Data type | Description |
|-----------------------|-----------|---|
| NLEAF | BIGINT | Total number of index leaf pages. |
| NUM_EMPTY_LEAFS | BIGINT | Number of pseudo-empty index leaf pages. |
| NLEVELS | INTEGER | Number of index levels. |
| NUMRIDS_DELETED | BIGINT | Number of pseudo-deleted RIDs. |
| FULLKEYCARD | BIGINT | Number of unique index entries that are not marked deleted. |
| 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 |
| 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. |

Related concepts:

- “SYSTOOLSPACE and SYSTOOLSTMPSPACE table spaces” in *Administration Guide: Planning*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “REORGCHK_TB_STATS procedure – Retrieve table statistics for reorganization evaluation” on page 553
- “REORGCHK command” in *Command Reference*

REORGCHK_IX_STATS

Related samples:

- "spclient.c -- Call various stored procedures"
- "SpClient.java -- Call a variety of types of stored procedures from SpServer.java (JDBC)"

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', specifies a fully qualified table name, or accepts one of the following values: ALL, USER, or SYSTEM. If *scope* has a value of 'S', specifies a schema name.

Authorization:

- SELECT privilege on catalog tables.
- EXECUTE privilege on the REORGCHK_TB_STATS procedure.

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 132. Information returned by the REORGCHK_TB_STATS procedure

| Column name | Data type | Description |
|--------------|--------------|---|
| TABLE_SCHEMA | VARCHAR(128) | Schema name. |
| TABLE_NAME | VARCHAR(128) | Table name. |
| CARD | BIGINT | Cardinality (number of rows in the table). |
| OVERFLOW | BIGINT | Number of overflow rows. |
| 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. |

REORGCHK_TB_STATS

Table 132. Information returned by the REORGCHK_TB_STATS procedure (continued)

| Column name | Data type | Description |
|---------------|-----------|---|
| 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) 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 |

Related concepts:

- “SYSTOOLSPACE and SYSTOOLSTMPSPACE table spaces” in *Administration Guide: Planning*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “REORGCHK_IX_STATS procedure – Retrieve index statistics for reorganization evaluation” on page 550
- “REORGCHK command” in *Command Reference*

Related samples:

- “spclient.c -- Call various stored procedures”
- “SpClient.java -- Call a variety of types of stored procedures from SpServer.java (JDBC)”

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 of type VARCHAR(32672) 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:

SQLERRM

EXECUTE privilege on the SQLERRM scalar function.

Examples:

Example 1: Retrieve the English short message for SQL0551N with tokens "AYYANG", "UPDATE" and "SYSCAT.TABLES".

```
VALUES (SYSPROC.SQLERRM
        ('SQL0551', '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 of type VARCHAR(32672) 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:

EXECUTE privilege on the SQLERRM scalar function.

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 "".
```

Related concepts:

- “Introduction to Messages” in *Message Reference Volume 1*

Related reference:

- “Supported administrative SQL routines and views” on page 8
- “SQLSTATE messages” in *Message Reference Volume 2*

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:

- 'AM' for creating activity monitor objects
- 'DB2AC' for autonomous computing (health monitor)
- 'STMG_DBSIZE_INFO' for storage management
- 'POLICY' for policy (tables and triggers)

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.

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 will be used if the tool name is AM. Otherwise, the SYSTOOLSPACE table space will be used. If SYSTOOLSPACE does not already exist, it will be created.

schema-name

Reserved for future use. The SYSTOOLS schema is always used regardless of the name passed into this parameter.

Example:

```

CALL SYSPROC.SYSINSTALLOBJECTS('AM', 'C', CAST (NULL AS VARCHAR(128)),
    CAST (NULL AS VARCHAR(128)))
    
```

Related concepts:

- “SYSTOOLSPACE and SYSTOOLSTMPSPACE table spaces” in *Administration Guide: Planning*

Related reference:

- “Supported administrative SQL routines and views” on page 8

Chapter 4. Deprecated administrative SQL routines

Deprecated SQL administrative routines and their replacement routines or views

In order to provide expanded support in DB2 Version 9 for the existing administrative routines, some of the DB2 UDB for Linux, UNIX, and Windows Version 8 routines have been replaced with new, more comprehensive routines or views.

Applications that use the DB2 UDB for Linux, UNIX, and Windows Version 8 table functions should be modified to use the new functions or administrative views. The new table functions have the same base names as the original functions but are suffixed with ‘_Vxx’ for the version of the product in which they were added (for example, _V91). In most cases, the new table functions and administrative views return additional information. The administrative views will always be based on the most current version of the table functions, and therefore allow for more application portability. Since the columns might vary from one release to the next (that is, some are added and some are deleted), it is recommended that specific columns be selected from the administrative views, or that the result set be described if a SELECT * statement is used by an application.

Table 133. Deprecated SQL administrative routines and their replacement routines or views

| DB2 UDB for Linux, UNIX, and Windows Version 8 deprecated function | New DB2 Version 9 function or view |
|---|---|
| “GET_DB_CONFIG ” on page 563 | “DBCFCG administrative view – Retrieve database configuration parameter information” on page 182 |
| “GET_DBM_CONFIG ” on page 565 | “DBMCFG administrative view – Retrieve database manager configuration parameter information” on page 184 |
| “SNAP_GET_CONTAINER ” on page 566 | “SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information” on page 351 |
| “SNAP_GET_DB ” on page 568 | “SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group” on page 356 |
| SNAP_GET_DETAILLOG (1) | “SNAPDETAILLOG administrative view and SNAP_GET_DETAILLOG_V91 table function – Retrieve snapshot information from the detail_log logical data group” on page 383 |
| “SNAP_GET_DYN_SQL ” on page 576 | “SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information” on page 387 |
| “SNAP_GET_STO_PATHS ” on page 579 | “SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS table function – Retrieve automatic storage path information” on page 421 |

Deprecated SQL administrative routines

Table 133. *Deprecated SQL administrative routines and their replacement routines or views (continued)*

| DB2 UDB for Linux, UNIX, and Windows Version 8 deprecated function | New DB2 Version 9 function or view |
|--|---|
| "SNAP_GET_TAB " on page 580 | "SNAPTAB administrative view and SNAP_GET_TAB_V91 table function – Retrieve table logical data group snapshot information" on page 432 |
| "SNAP_GET_TBSP " on page 582 | "SNAPTbsp administrative view and SNAP_GET_TBSP_V91 table function – Retrieve tablespace logical data group snapshot information" on page 441 |
| "SNAP_GET_TBSP_PART " on page 586 | "SNAPTbsp_PART administrative view and SNAP_GET_TBSP_PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information" on page 447 |
| "SNAPSHOT_AGENT " on page 589 | "SNAPAGENT administrative view and SNAP_GET_AGENT table function – Retrieve agent logical data group application snapshot information" on page 315 |
| "SNAPSHOT_APPL " on page 590 | "SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information" on page 324 |
| "SNAPSHOT_APPL_INFO " on page 596 | "SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information" on page 334 |
| "SNAPSHOT_BP " on page 599 | "SNAPBP administrative view and SNAP_GET_BP table function – Retrieve bufferpool logical group snapshot information" on page 341 |
| "SNAPSHOT_CONTAINER " on page 602 | "SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information" on page 351 |
| "SNAPSHOT_DATABASE " on page 604 | "SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group" on page 356 |
| "SNAPSHOT_DBM " on page 611 | "SNAPDBM administrative view and SNAP_GET_DBM table function – Retrieve the dbm logical grouping snapshot information" on page 374 |
| "SNAPSHOT_DYN_SQL " on page 614 | "SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information" on page 387 |
| "SNAPSHOT_FCM " on page 616 | "SNAPFCM administrative view and SNAP_GET_FCM table function – Retrieve the fcm logical data group snapshot information" on page 392 |
| "SNAPSHOT_FCMNODE " on page 618 | "SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function – Retrieve the fcm_node logical data group snapshot information" on page 395 |
| "SNAPSHOT_FILEW " on page 619 | "SNAP_WRITE_FILE procedure" on page 313 |

Table 133. *Deprecated SQL administrative routines and their replacement routines or views (continued)*

| DB2 UDB for Linux, UNIX, and Windows Version 8 deprecated function | New DB2 Version 9 function or view |
|--|--|
| "SNAPSHOT_LOCK " on page 620 | "SNAPLOCK administrative view and SNAP_GET_LOCK table function – Retrieve lock logical data group snapshot information" on page 403 |
| "SNAPSHOT_LOCKWAIT " on page 622 | "SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function – Retrieve lockwait logical data group snapshot information" on page 409 |
| "SNAPSHOT QUIESCERS " on page 624 | "SNAPTbsp_QUIESCER administrative view and SNAP_GET_TBSP_QUIESCER table function – Retrieve quiescer table space snapshot information" on page 452 |
| "SNAPSHOT_RANGES " on page 626 | "SNAPTbsp_RANGE administrative view and SNAP_GET_TBSP_RANGE table function – Retrieve range snapshot information" on page 456 |
| "SNAPSHOT_STATEMENT " on page 628 | "SNAPSTMT administrative view and SNAP_GET_STMT table function – Retrieve statement snapshot information" on page 415 |
| "SNAPSHOT_SUBSECT " on page 631 | "SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function – Retrieve subsection logical monitor group snapshot information" on page 425 |
| "SNAPSHOT_SWITCHES " on page 633 | "SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function – Retrieve database snapshot switch state information" on page 429 |
| "SNAPSHOT_TABLE " on page 635 | "SNAPTAB administrative view and SNAP_GET_TAB_V91 table function – Retrieve table logical data group snapshot information" on page 432 |
| "SNAPSHOT_TBREORG " on page 637 | "SNAPTAB_REORG administrative view and SNAP_GET_TAB_REORG table function – Retrieve table reorganization snapshot information" on page 436 |
| "SNAPSHOT_TBS " on page 639 | "SNAPTbsp administrative view and SNAP_GET_TBSP_V91 table function – Retrieve tablespace logical data group snapshot information" on page 441 |
| "SNAPSHOT_TBS_CFG " on page 642 | "SNAPTbsp_PART administrative view and SNAP_GET_TBSP_PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information" on page 447 |
| SNAPSHOT_UTIL (1) | "SNAPUTIL administrative view and SNAP_GET_UTIL table function – Retrieve utility_info logical data group snapshot information" on page 460 |
| SNAPSHOT_UTIL_PROG (1) | "SNAPUTIL_PROGRESS administrative view and SNAP_GET_UTIL_PROGRESS table function – Retrieve progress logical data group snapshot information" on page 464 |

Deprecated SQL administrative routines

Table 133. *Deprecated SQL administrative routines and their replacement routines or views (continued)*

| DB2 UDB for Linux, UNIX, and Windows Version 8 deprecated function | New DB2 Version 9 function or view |
|--|---|
| "SQLCACHE_SNAPSHOT " on page 645 | "SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information" on page 387. Information returned by the SQLCACHE_SNAPSHOT table function is now included in this new view and table function. |
| SYSFUN.GROUPS (1) | This procedure has been deprecated. |
| SYSFUN.GROUPS_FOR_USER (1) | "AUTH_LIST_GROUPS_FOR_AUTHID table function – Retrieve group membership list for a given authorization ID" on page 273 |
| SYSFUN.USER_GROUPS (1) | This procedure has been deprecated. |
| SYSFUN.USERS (1) | This procedure has been deprecated. |
| "SYSINSTALLROUTINES " on page 647 | This procedure has been deprecated. |

Note: (1) These functions were present in DB2 UDB for Linux, UNIX, and Windows Version 8, but were omitted from the documentation.

Related reference:

- "Supported administrative SQL routines and views" on page 8

GET_DB_CONFIG

Note: This procedure has been deprecated and replaced by the “DBCFCG administrative view – Retrieve database configuration parameter information” on page 182.

►►—GET_DB_CONFIG—(—)—————◄◄

The schema is SYSPROC.

The GET_DB_CONFIG procedure returns database configuration information. The procedure does not take any arguments.

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 below.

Table 134. Information returned by the GET_DB_CONFIG procedure

| Column name | Data type | Description |
|---------------|-----------|---|
| DBCONFIG_TYPE | 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.

Example

Using the command line processor (CLP), change the value of the *logretain* and the *userexit* database configuration parameters. 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 LOGRETAIN RECOVERY USEREXIT ON

CALL SYSPROC.GET_DB_CONFIG()

SELECT DBCONFIG_TYPE, LOGRETAIN, USEREXIT
      FROM SESSION.DB_CONFIG

CONNECT RESET
```

The following is an example of output from this query.

GET_DB_CONFIG

| DBCONFIG_TYPE | LOGRETAIN | USEREXIT |
|---------------|-----------|----------|
| 0 | 1 | 1 |
| 1 | 0 | 0 |

2 record(s) selected.

Related reference:

- "Configuration parameters summary" in *Performance Guide*
- "GET_DBM_CONFIG " on page 565

GET_DBM_CONFIG

Note: This table function has been deprecated and replaced by the “DBMCFG administrative view – Retrieve database manager configuration parameter information” on page 184.

►►—GET_DBM_CONFIG—(—)—————►►

The schema is SYSFUN.

The GET_DBM_CONFIG table function returns database manager configuration information. The function does not take any arguments.

The function returns a table with two rows containing a column for each parameter. The first column is named DBMCONFIG_TYPE, as shown below.

Table 135. 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.

Related reference:

- “Configuration parameters summary” in *Performance Guide*
- “GET_DB_CONFIG ” on page 563
- “Supported administrative SQL routines and views” on page 8

SNAP_GET_CONTAINER

Note: This table function has been deprecated and replaced by the “SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information” on page 351

▶▶—SNAP_GET_CONTAINER—(—dbname—, —dbpartitionnum—)————▶▶

The schema is SYSPROC.

The SNAP_GET_CONTAINER table function returns snapshot information from the tablespace_container logical data group.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 136. Information returned by 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 monitor element |
| TBSP_ID | BIGINT | tablespace_id - Table Space Identification monitor element |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container Name monitor element |
| CONTAINER_ID | BIGINT | container_id - Container Identification monitor element |
| CONTAINER_TYPE | SMALLINT | container_type - Container Type monitor element |
| TOTAL_PAGES | BIGINT | container_total_pages - Total Pages in Container monitor element |

Table 136. Information returned by the SNAP_GET_CONTAINER table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------|-----------|--|
| USABLE_PAGES | BIGINT | container_usable_pages - Usable Pages in Container monitor element |
| ACCESSIBLE | SMALLINT | container_accessible - Accessibility of Container monitor element |
| STRIPE_SET | BIGINT | container_stripe_set - Stripe Set monitor element |
| DBPARTITIONNUM | SMALLINT | node_number - Node Number monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAP_GET_DB

Note: This table function has been deprecated and replaced by the “SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group” on page 356

▶▶—SNAP_GET_DB—(—dbname—, —dbpartitionnum—)—————▶▶

The schema is SYSPROC.

The SNAP_GET_DB table function returns snapshot information from the database.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 137. Information returned by the 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 monitor element |
| DB_PATH | VARCHAR(1024) | db_path - Database Path monitor element |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input Database Alias monitor element |
| DB_STATUS | BIGINT | db_status - Status of Database monitor element |
| CATALOG_PARTITION | SMALLINT | catalog_node - Catalog Node Number monitor element |
| CATALOG_PARTITION_NAME | VARCHAR(128) | catalog_node_name - Catalog Node Network Name monitor element |
| SERVER_PLATFORM | INTEGER | server_platform - Server Operating System monitor element |
| DB_LOCATION | INTEGER | db_location - Database Location monitor element |

Table 137. Information returned by the SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------|-----------|---|
| DB_CONN_TIME | TIMESTAMP | db_conn_time - Database Activation Timestamp monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |
| 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 |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum Number of Coordinating Agents monitor element |
| LOCKS_HELD | BIGINT | locks_held - Locks Held monitor element |
| LOCK_WAITS | BIGINT | lock_waits - Lock Waits monitor element |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time Waited On Locks monitor element |
| LOCK_LIST_IN_USE | BIGINT | lock_list_in_use - Total Lock List Memory In Use monitor element |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks Detected monitor element |
| LOCK_ESCALS | BIGINT | lock_escals - Number of Lock Escalations monitor element |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive Lock Escalations monitor element |
| LOCKS_WAITING | BIGINT | locks_waiting - Current Agents Waiting On Locks monitor element |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of Lock Timeouts monitor element |

Table 137. Information returned by the SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| NUM_INDOUBT_TRANS | BIGINT | num_indoubt_trans - Number of Indoubt Transactions monitor element |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total Sort Heap Allocated 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 Water Mark monitor element |
| TOTAL_SORTS | BIGINT | total_sorts - Total Sorts monitor element |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total Sort Time monitor element |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort Overflows monitor element |
| ACTIVE_SORTS | BIGINT | active_sorts - Active Sorts monitor element |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads monitor element |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads monitor element |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer Pool Asynchronous Data Reads monitor element |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer Pool Data Writes monitor element |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer Pool Asynchronous Data Writes monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads monitor element |

Table 137. Information returned by the SNAP_GET_DB 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 monitor element |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer Pool Index Writes monitor element |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer Pool Asynchronous Index Reads monitor element |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer Pool Asynchronous Index 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 |
| 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 |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests monitor element |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests monitor element |
| 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 |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time Waited for Prefetch monitor element |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread Prefetch Pages monitor element |
| DIRECT_READS | BIGINT | direct_reads - Direct Reads From Database monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct Writes to Database monitor element |

Table 137. Information returned by the SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding 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 |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct Read Time monitor element |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct Write Time monitor element |
| FILES_CLOSED | BIGINT | files_closed - Database Files Closed monitor element |
| POOL_DATA_TO_ESTORE | BIGINT | pool_data_to_estore - Buffer Pool Data Pages to Extended Storage monitor element |
| POOL_INDEX_TO_ESTORE | BIGINT | pool_index_to_estore - Buffer Pool Index Pages to Extended Storage monitor element |
| POOL_INDEX_FROM_ESTORE | BIGINT | pool_index_from_estore - Buffer Pool Index Pages from Extended Storage monitor element |
| POOL_DATA_FROM_ESTORE | BIGINT | pool_data_from_estore - Buffer Pool Data Pages from Extended Storage monitor element |
| ELAPSED_EXEC_TIME_S | BIGINT | elapsed_exec_time - Statement Execution Elapsed Time monitor element |
| ELAPSED_EXEC_TIME_MS | BIGINT | elapsed_exec_time - Statement Execution Elapsed Time monitor element |
| COMMIT_SQL_STMTS | BIGINT | commit_sql_stmts - Commit Statements Attempted monitor element |
| ROLLBACK_SQL_STMTS | BIGINT | rollback_sql_stmts - Rollback Statements Attempted 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 |

Table 137. Information returned by the SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|-----------|---|
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| INT_AUTO_REBINDS | BIGINT | int_auto_rebinds - Internal Automatic Rebinds 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 |
| INT_COMMITS | BIGINT | int_commits - Internal Commits monitor element |
| INT_ROLLBACKS | BIGINT | int_rollback - Internal Rollbacks monitor element |
| INT_DEADLOCK_ROLLBACKS | BIGINT | int_deadlock_rollback - Internal Rollbacks Due To Deadlock 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 |
| ROWS_SELECTED | BIGINT | rows_selected - Rows Selected monitor element |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/Precompiles Attempted 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 |

Table 137. Information returned by the SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|-----------|--|
| LOG_READ_TIME_S | BIGINT | log_read_time - Log Read Time monitor element |
| LOG_READ_TIME_NS | BIGINT | log_read_time - Log Read Time monitor element |
| LOG_WRITES | BIGINT | log_writes - Number of Log Pages Written monitor element |
| LOG_WRITE_TIME_S | BIGINT | log_write_time - Log Write Time monitor element |
| LOG_WRITE_TIME_NS | 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 Full Log Buffers monitor element |
| NUM_LOG_DATA_FOUND_IN_BUFFER | BIGINT | num_log_data_found_in_buffer - Number of Log Data Found In Buffer monitor element |
| APPL_ID_OLDEST_XACT | BIGINT | appl_id_oldest_xact - Application with Oldest Transaction monitor element |
| 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 |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package Cache Lookups monitor element |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package Cache Inserts monitor element |
| PKG_CACHE_NUM_OVERFLOWS | BIGINT | pkg_cache_num_overflows - Package Cache Overflows monitor element |
| PKG_CACHE_SIZE_TOP | BIGINT | pkg_cache_size_top - Package Cache High Water Mark monitor element |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - Section Lookups monitor element |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Section Inserts monitor element |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog Cache Lookups monitor element |

Table 137. Information returned by the SNAP_GET_DB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------------|-----------|--|
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog Cache Inserts monitor element |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog Cache Overflows monitor element |
| CAT_CACHE_SIZE_TOP | BIGINT | cat_cache_size_top - Catalog Cache High Water Mark monitor element |
| PRIV_WORKSPACE_SIZE_TOP | BIGINT | priv_workspace_size_top - Maximum Private Workspace Size monitor element |
| PRIV_WORKSPACE_NUM_OVERFLOWS | BIGINT | priv_workspace_num_overflows - Private Workspace Overflows monitor element |
| PRIV_WORKSPACE_SECTION_INSERTS | BIGINT | priv_workspace_section_inserts - Private Workspace Section Inserts monitor element |
| PRIV_WORKSPACE_SECTION_LOOKUPS | BIGINT | priv_workspace_section_lookups - Private Workspace Section Lookups monitor element |
| SHR_WORKSPACE_SIZE_TOP | BIGINT | shr_workspace_size_top - Maximum Shared Workspace Size monitor element |
| SHR_WORKSPACE_NUM_OVERFLOWS | BIGINT | shr_workspace_num_overflows - Shared Workspace Overflows monitor element |
| SHR_WORKSPACE_SECTION_INSERTS | BIGINT | shr_workspace_section_inserts - Shared Workspace Section Inserts monitor element |
| SHR_WORKSPACE_SECTION_LOOKUPS | BIGINT | shr_workspace_section_lookups - Shared Workspace Section Lookups 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 |
| NUM_DB_STORAGE_PATHS | BIGINT | num_db_storage_paths - Number of automatic storage paths monitor element |
| DBPARTITIONNUM | SMALLINT | node_number - Node Number monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAP_GET_DYN_SQL

Note: This table function has been deprecated and replaced by the “SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information” on page 387

▶▶—SNAP_GET_DYN_SQL—(—dbname—, —dbpartitionnum—)—————▶▶

The schema is SYSPROC.

The SNAP_GET_DYN_SQL table function returns snapshot information from the dynsql logical data group.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 138. Information returned by 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 monitor element |
| NUM_COMPILATIONS | BIGINT | num_compilations - Statement Compilations monitor element |
| PREP_TIME_WORST | BIGINT | prep_time_worst - Statement Worst Preparation Time monitor element |
| PREP_TIME_BEST | BIGINT | prep_time_best - Statement Best Preparation Time 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 |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |

Table 138. Information returned by the SNAP_GET_DYN_SQL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|--|
| ROWS_WRITTEN | BIGINT | rows_written - Rows Written monitor element |
| STMT_SORTS | BIGINT | stmt_sorts - Statement Sorts monitor element |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort Overflows monitor element |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total Sort Time monitor element |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads monitor element |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads monitor element |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads monitor element |
| TOTAL_EXEC_TIME | BIGINT | total_exec_time - Elapsed Statement Execution Time monitor element |
| TOTAL_EXEC_TIME_MS | BIGINT | total_exec_time - Elapsed Statement Execution Time monitor element |
| TOTAL_USR_TIME | BIGINT | total_usr_cpu_time - Total User CPU for a Statement monitor element |
| TOTAL_USR_TIME_MS | BIGINT | total_usr_cpu_time - Total User CPU for a Statement monitor element |
| TOTAL_SYS_TIME | BIGINT | total_sys_cpu_time - Total System CPU for a Statement monitor element |
| TOTAL_SYS_TIME_MS | BIGINT | total_sys_cpu_time - Total System CPU for a Statement monitor element |

SNAP_GET_DYN_SQL

Table 138. Information returned by the SNAP_GET_DYN_SQL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------|-----------|---|
| STMT_TEXT | CLOB | stmt_text - SQL Dynamic Statement Text monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAP_GET_STO_PATHS

Note: This table function has been deprecated and replaced by the “SNAPSTORAGE_PATHS administrative view and SNAP_GET_STORAGE_PATHS table function – Retrieve automatic storage path information” on page 421

►►—SNAP_GET_STO_PATHS—(—dbname—,—dbpartitionnum—)—————►►

The schema is SYSPROC.

The SNAP_GET_STO_PATHS table function returns snapshot information from the storage_paths logical data group.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 139. Information returned by the SNAP_GET_STO_PATHS 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 monitor element |
| DB_STORAGE_PATH | VARCHAR(256) | db_storage_path - Automatic storage path monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAP_GET_TAB

Note: This table function has been deprecated and replaced by the “SNAPTAB administrative view and SNAP_GET_TAB_V91 table function – Retrieve table logical data group snapshot information” on page 432

▶▶—SNAP_GET_TAB—(—*dbname*—,—*dbpartitionnum*—)—▶▶

The schema is SYSPROC.

The SNAP_GET_TAB table function returns snapshot information from the table logical data group.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 140. Information returned by 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. |
| TABSHEMA | VARCHAR(128) | table_schema - Table Schema Name monitor element |
| TABNAME | VARCHAR(128) | table_name - Table Name monitor element |
| TAB_FILE_ID | BIGINT | table_file_id - Table File ID monitor element |
| TAB_TYPE | BIGINT | table_type - Table Type monitor element |
| DATA_OBJECT_PAGES | BIGINT | data_object_pages - Data Object Pages monitor element |
| INDEX_OBJECT_PAGES | BIGINT | index_object_pages - Index Object Pages monitor element |
| LOB_OBJECT_PAGES | BIGINT | lob_object_pages - LOB Object Pages monitor element |

Table 140. Information returned by the SNAP_GET_TAB table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------|-----------|--|
| LONG_OBJECT_PAGES | BIGINT | long_object_pages - Long Object Pages monitor element |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| ROWS_WRITTEN | BIGINT | rows_written - Rows Written monitor element |
| OVERFLOW_ACCESSES | BIGINT | overflow_accesses - Accesses to Overflowed Records monitor element |
| PAGE_REORGS | BIGINT | page_reorgs - Page Reorganizations monitor element |
| DBPARTITIONNUM | SMALLINT | node_number - Node Number monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAP_GET_TBSP

Note: This table function has been deprecated and replaced by the “SNAPTbsp administrative view and SNAP_GET_TBSP_V91 table function – Retrieve tablespace logical data group snapshot information” on page 441

▶▶—SNAP_GET_TBSP—(—dbname—,—dbpartitionnum—)—————▶▶

The schema is SYSPROC.

The SNAP_GET_TBSP table function returns snapshot information from the tablespace logical data group.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 141. Information returned by 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 monitor element |
| TBSP_ID | BIGINT | tablespace_id - Table Space Identification monitor element |
| TBSP_TYPE | SMALLINT | tablespace_type - Table Space Type monitor element |
| TBSP_CONTENT_TYPE | SMALLINT | tablespace_content_type - Table Space Contents Type monitor element |
| TBSP_PAGE_SIZE | BIGINT | tablespace_page_size - Table Space Page Size monitor element |
| TBSP_EXTENT_SIZE | BIGINT | tablespace_extent_size - Table Space Extent Size monitor element |
| TBSP_PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table Space Prefetch Size monitor element |

Table 141. Information returned by the SNAP_GET_TBSP table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|---|
| TBSP_CUR_POOL_ID | BIGINT | tablespace_cur_pool_id - Buffer Pool Currently Being Used monitor element |
| TBSP_NEXT_POOL_ID | BIGINT | tablespace_next_pool_id - Buffer Pool That Will Be Used at Next Startup monitor element |
| FS_CACHING ¹ | SMALLINT | fs_caching - File System Caching monitor element |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |
| POOL_TEMP_DATA_L_READS | BIGINT | pool_temp_data_l_reads - Buffer Pool Temporary Data Logical Reads monitor element |
| POOL_TEMP_DATA_P_READS | BIGINT | pool_temp_data_p_reads - Buffer Pool Temporary Data Physical Reads monitor element |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer Pool Asynchronous Data Reads monitor element |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer Pool Data Writes monitor element |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer Pool Asynchronous Data Writes monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |
| POOL_TEMP_INDEX_L_READS | BIGINT | pool_temp_index_l_reads - Buffer Pool Temporary Index Logical Reads monitor element |
| POOL_TEMP_INDEX_P_READS | BIGINT | pool_temp_index_p_reads - Buffer Pool Temporary Index Physical Reads monitor element |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer Pool Asynchronous Index Reads monitor element |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer Pool Index Writes monitor element |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer Pool Asynchronous Index Writes monitor element |

SNAP_GET_TBSP

Table 141. Information returned by the SNAP_GET_TBSP table function (continued)

| Column name | Data type | Description or corresponding 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 |
| 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 |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests monitor element |
| POOL_ASYNC_INDEX_READ_REQS | BIGINT | pool_async_index_read_reqs - Buffer Pool Asynchronous Index Read Requests monitor element |
| POOL_NO_VICTIM_BUFFER | BIGINT | pool_no_victim_buffer - Buffer Pool No Victim Buffers monitor element |
| DIRECT_READS | BIGINT | direct_reads - Direct Reads From Database monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct Writes to Database 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 |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct Read Time monitor element |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct Write Time monitor element |
| FILES_CLOSED | BIGINT | files_closed - Database Files Closed monitor element |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread Prefetch Pages monitor element |
| POOL_DATA_TO_ESTORE | BIGINT | pool_data_to_estore - Buffer Pool Data Pages to Extended Storage monitor element |
| POOL_INDEX_TO_ESTORE | BIGINT | pool_index_to_estore - Buffer Pool Index Pages to Extended Storage monitor element |
| POOL_INDEX_FROM_ESTORE | BIGINT | pool_index_from_estore - Buffer Pool Index Pages from Extended Storage monitor element |
| POOL_DATA_FROM_ESTORE | BIGINT | pool_data_from_estore - Buffer Pool Data Pages from Extended Storage monitor element |

Table 141. Information returned by the SNAP_GET_TBSP table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------------|-----------|---|
| TBSP_REBALANCER_MODE | BIGINT | tablespace_rebalancer_mode - Rebalancer Mode monitor element |
| TBSP_USING_AUTO_STORAGE | SMALLINT | tablespace_using_auto_storage - Using automatic storage monitor element |
| TBSP_AUTO_RESIZE_ENABLED | SMALLINT | tablespace_auto_resize_enabled - Auto-resize enabled monitor element |

¹ If FS_CACHING is 0, file system caching is enabled, and if FS_CACHING is 1, file system caching is disabled.

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAP_GET_TBSP_PART

Note: This table function has been deprecated and replaced by the “SNAPTbsp_Part administrative view and SNAP_GET_TBSP_PART_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information” on page 447

▶▶—SNAP_GET_TBSP_PART—(—dbname—,—dbpartitionnum—)————▶▶

The schema is SYSPROC.

The SNAP_GET_TBSP_PART table function returns snapshot information from the tablespace_nodeinfo logical data group.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 142. Information returned by 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 monitor element |
| TBSP_ID | BIGINT | tablespace_id - Table Space Identification monitor element |
| TBSP_STATE | BIGINT | tablespace_state - Table Space State monitor element |
| TBSP_PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table Space Prefetch Size monitor element |
| TBSP_NUM_QUIESCERS | BIGINT | tablespace_num_quiescers - Number of Quiescers monitor element |
| TBSP_STATE_CHANGE_OBJECT_ID | BIGINT | tablespace_state_change_object_id - State Change Object Identification monitor element |

Table 142. Information returned by the SNAP_GET_TBSP_PART table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|-----------|---|
| TBSP_STATE_CHANGE_TBSP_ID | BIGINT | tablespace_state_change_ts_id - State Change Table Space Identification monitor element |
| TBSP_MIN_RECOVERY_TIME | TIMESTAMP | tablespace_min_recovery_time - Minimum Recovery Time For Rollforward monitor element |
| TBSP_TOTAL_PAGES | BIGINT | tablespace_total_pages - Total Pages in Table Space monitor element |
| TBSP_USABLE_PAGES | BIGINT | tablespace_usable_pages - Usable Pages in Table Space monitor element |
| TBSP_USED_PAGES | BIGINT | tablespace_used_pages - Used Pages in Table Space monitor element |
| TBSP_FREE_PAGES | BIGINT | tablespace_free_pages - Free Pages in Table Space monitor element |
| TBSP_PENDING_FREE_PAGES | BIGINT | tablespace_pending_free_pages - Pending Free Pages in Table Space monitor element |
| TBSP_PAGE_TOP | BIGINT | tablespace_page_top - Table Space High Water Mark monitor element |
| REBALANCER_MODE | BIGINT | tablespace_rebalancer_mode - Rebalancer Mode monitor element |
| REBALANCER_EXTENTS_REMAINING | BIGINT | tablespace_rebalancer_extents_remaining - Total Number of Extents to be Processed by the Rebalancer monitor element |
| REBALANCER_EXTENTS_PROCESSED | BIGINT | tablespace_rebalancer_extents_processed - Number of Extents the Rebalancer has Processed monitor element |
| REBALANCER_PRIORITY | BIGINT | tablespace_rebalancer_priority - Current Rebalancer Priority monitor element |
| REBALANCER_START_TIME | TIMESTAMP | tablespace_rebalancer_start_time - Rebalancer Start Time monitor element |
| REBALANCER_RESTART_TIME | TIMESTAMP | tablespace_rebalancer_restart_time - Rebalancer Restart Time monitor element |
| REBALANCER_LAST_EXTENT_MOVED | BIGINT | tablespace_rebalancer_last_extent_moved - Last Extent Moved by the Rebalancer monitor element |
| TBSP_NUM_RANGES | BIGINT | tablespace_num_ranges - Number of Ranges in the Table Space Map monitor element |
| TBSP_NUM_CONTAINERS | BIGINT | tablespace_num_containers - Number of Containers in Table Space monitor element |
| TBSP_INITIAL_SIZE | BIGINT | tablespace_initial_size - Initial table space size monitor element |
| TBSP_CURRENT_SIZE | BIGINT | tablespace_current_size - Current table space size monitor element |
| TBSP_MAX_SIZE | BIGINT | tablespace_max_size - Maximum table space size monitor element |
| TBSP_INCREASE_SIZE | BIGINT | tablespace_increase_size - Increase size in bytes monitor element |

SNAP_GET_TBSP_PART

Table 142. Information returned by the SNAP_GET_TBSP_PART table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------------|-----------|--|
| TBSP_INCREASE_SIZE_PERCENT | SMALLINT | tablespace_increase_size_percent - Increase size by percent monitor element |
| TBSP_LAST_RESIZE_TIME | TIMESTAMP | tablespace_last_resize_time - Time of last successful resize monitor element |
| TBSP_LAST_RESIZE_FAILED | SMALLINT | tablespace_last_resize_failed - Last resize attempt failed monitor element |
| DBPARTITIONNUM | SMALLINT | node_number - Node Number monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_AGENT

Note: This table function has been deprecated and replaced by the “SNAPAGENT administrative view and SNAP_GET_AGENT table function – Retrieve agent logical data group application snapshot information” on page 315.

▶▶—SNAPSHOT_AGENT—(—dbname—,—dbpartitionnum—)————▶▶

The schema is SYSPROC.

The SNAPSHOT_AGENT function returns information about agents from an application snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 143. Information returned by the SNAPSHOT_AGENT 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) monitor element |
| AGENT_PID | BIGINT | agent_pid - Process or Thread ID monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_APPL

Note: This table function has been deprecated and replaced by the “SNAPAPPL administrative view and SNAP_GET_APPL table function – Retrieve appl logical data group snapshot information” on page 324.

▶▶—SNAPSHOT_APPL—(—dbname—,—dbpartitionnum—)—————▶▶

The schema is SYSPROC.

The SNAPSHOT_APPL function returns general information from an application snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 144. Information returned by the SNAPSHOT_APPL 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) monitor element |
| UOW_LOG_SPACE_USED | BIGINT | uow_log_space_used - Unit of Work Log Space Used monitor element |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| ROWS_WRITTEN | BIGINT | rows_written - Rows Written monitor element |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |

Table 144. Information returned by the SNAPSHOT_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|-----------|--|
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer Pool Data Writes monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer Pool Index 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 |
| DIRECT_READS | BIGINT | direct_reads - Direct Reads From Database monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct Writes to Database 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 |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct Read Time monitor element |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct Write Time monitor element |
| POOL_DATA_TO_ESTORE | BIGINT | pool_data_to_estore - Buffer Pool Data Pages to Extended Storage monitor element |
| POOL_INDEX_TO_ESTORE | BIGINT | pool_index_to_estore - Buffer Pool Index Pages to Extended Storage monitor element |
| POOL_INDEX_FROM_ESTORE | BIGINT | pool_index_from_estore - Buffer Pool Index Pages from Extended Storage monitor element |
| POOL_DATA_FROM_ESTORE | BIGINT | pool_data_from_estore - Buffer Pool Data Pages from Extended Storage monitor element |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread Prefetch Pages monitor element |
| LOCKS_HELD | BIGINT | locks_held - Locks Held monitor element |
| LOCK_WAITS | BIGINT | lock_waits - Lock Waits monitor element |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time Waited On Locks monitor element |

Table 144. Information returned by the SNAPSHOT_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|---|
| LOCK_ESCALS | BIGINT | lock_escals - Number of Lock Escalations monitor element |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive Lock Escalations monitor element |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks Detected monitor element |
| TOTAL_SORTS | BIGINT | total_sorts - Total Sorts monitor element |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total Sort Time monitor element |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort Overflows monitor element |
| COMMIT_SQL_STMTS | BIGINT | commit_sql_stmts - Commit Statements Attempted monitor element |
| ROLLBACK_SQL_STMTS | BIGINT | rollback_sql_stmts - Rollback Statements Attempted 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 |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| INT_AUTO_REBINDS | BIGINT | int_auto_rebinds - Internal Automatic Rebinds monitor element |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal Rows Updated monitor element |
| INT_COMMITS | BIGINT | int_commits - Internal Commits monitor element |
| INT_ROLLBACKS | BIGINT | int_rollbacks - Internal Rollbacks monitor element |

Table 144. Information returned by the SNAPSHOT_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|-----------|--|
| INT_DEADLOCK_ROLLBACKS | BIGINT | int_deadlock_rollback - Internal Rollbacks Due To Deadlock 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 |
| ROWS_SELECTED | BIGINT | rows_selected - Rows Selected monitor element |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/Precompiles Attempted monitor element |
| OPEN_REM_CURS | BIGINT | open_rem_curs - Open Remote Cursors monitor element |
| OPEN_REM_CURS_BLK | BIGINT | open_rem_curs_blk - Open Remote Cursors with Blocking monitor element |
| REJ_CURS_BLK | BIGINT | rej_curs_blk - Rejected Block Cursor Requests monitor element |
| ACC_CURS_BLK | BIGINT | acc_curs_blk - Accepted Block Cursor Requests monitor element |
| SQL_REQS_SINCE_COMMIT | BIGINT | sql_reqs_since_commit - SQL Requests Since Last Commit monitor element |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of Lock Timeouts monitor element |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal Rows Inserted monitor element |
| OPEN_LOC_CURS | BIGINT | open_loc_curs - Open Local Cursors monitor element |
| OPEN_LOC_CURS_BLK | BIGINT | open_loc_curs_blk - Open Local Cursors with Blocking monitor element |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package Cache Lookups monitor element |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package Cache Inserts monitor element |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog Cache Lookups monitor element |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog Cache Inserts monitor element |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog Cache Overflows monitor element |
| CAT_CACHE_HEAP_FULL | BIGINT | cat_cache_overflows - Catalog Cache Overflows monitor element |

SNAPSHOT_APPL

Table 144. Information returned by the SNAPSHOT_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-----------|--|
| NUM_AGENTS | BIGINT | num_agents - Number of Agents Working on a Statement monitor element |
| AGENTS_STOLEN | BIGINT | agents_stolen - Stolen Agents monitor element |
| ASSOCIATED_AGENTS_TOP | BIGINT | associated_agents_top - Maximum Number of Associated Agents monitor element |
| APPL_PRIORITY | BIGINT | appl_priority - Application Agent Priority monitor element |
| APPL_PRIORITY_TYPE | BIGINT | appl_priority_type - Application Priority Type monitor element |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time Waited for Prefetch monitor element |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - Section Lookups monitor element |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Section Inserts monitor element |
| LOCKS_WAITING | BIGINT | locks_waiting - Current Agents Waiting On Locks 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 |
| APPL_IDLE_TIME | BIGINT | appl_idle_time - Application Idle Time monitor element |
| UOW_LOCK_WAIT_TIME | BIGINT | uow_lock_wait_time - Total Time Unit of Work Waited on Locks monitor element |
| UOW_COMP_STATUS | BIGINT | uow_comp_status - Unit of Work Completion Status monitor element |
| AGENT_USR_CPU_TIME_S | BIGINT | agent_usr_cpu_time - User CPU Time used by Agent monitor element |
| AGENT_USR_CPU_TIME_MS | BIGINT | agent_usr_cpu_time - User CPU Time used by Agent monitor element |
| AGENT_SYS_CPU_TIME_S | BIGINT | agent_sys_cpu_time - System CPU Time used by Agent monitor element |
| AGENT_SYS_CPU_TIME_MS | BIGINT | agent_sys_cpu_time - System CPU Time used by Agent monitor element |

Table 144. Information returned by the SNAPSHOT_APPL table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|-------------|---|
| APPL_CON_TIME | TIMESTAMP | appl_con_time - Connection Request Start Timestamp monitor element |
| CONN_COMPLETE_TIME | TIMESTAMP | conn_complete_time - Connection Request Completion Timestamp monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp 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_ELAPSED_TIME_S | BIGINT | uow_elapsed_time - Most Recent Unit of Work Elapsed Time monitor element |
| UOW_ELAPSED_TIME_MS | BIGINT | uow_elapsed_time - Most Recent Unit of Work Elapsed Time monitor element |
| ELAPSED_EXEC_TIME_S | BIGINT | elapsed_exec_time - Statement Execution Elapsed Time monitor element |
| ELAPSED_EXEC_TIME_MS | BIGINT | elapsed_exec_time - Statement Execution Elapsed Time monitor element |
| INBOUND_COMM_ADDRESS | VARCHAR(32) | inbound_comm_address - Inbound Communication Address monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_APPL_INFO

Note: This table function has been deprecated and replaced by the “SNAPAPPL_INFO administrative view and SNAP_GET_APPL_INFO table function – Retrieve appl_info logical data group snapshot information” on page 334.

▶▶—SNAPSHOT_APPL_INFO—(—dbname—,—dbpartitionnum—)————▶▶

The schema is SYSPROC.

The SNAPSHOT_APPL_INFO function returns general information from an application snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 145. Information returned by the SNAPSHOT_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) monitor element |
| APPL_STATUS | BIGINT | appl_status - Application Status monitor element |
| CODEPAGE_ID | BIGINT | codepage_id - ID of Code Page Used by Application monitor element |
| NUM_ASSOC_AGENTS | BIGINT | num_assoc_agents - Number of Associated Agents monitor element |
| COORD_PARTITION_NUM | BIGINT | coord_node - Coordinating Node monitor element |

Table 145. Information returned by the SNAPSHOT_APPL_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------------|------------------|--|
| AUTHORITY_LVL | BIGINT | authority_lvl - User Authorization Level monitor element |
| CLIENT_PID | BIGINT | client_pid - Client Process ID monitor element |
| COORD_AGENT_PID | BIGINT | coord_agent_pid - Coordinator Agent monitor element |
| STATUS_CHANGE_TIME | TIMESTAMP | status_change_time - Application Status Change Time monitor element |
| CLIENT_PLATFORM | SMALLINT | client_platform - Client Operating Platform monitor element |
| CLIENT_PROTOCOL | SMALLINT | client_protocol - Client Communication Protocol monitor element |
| COUNTRY_CODE | SMALLINT | territory_code - Database Territory Code monitor element |
| APPL_NAME | VARCHAR(256) | appl_name - Application Name monitor element |
| APPL_ID | VARCHAR(128) | appl_id - Application ID monitor element |
| SEQUENCE_NO | VARCHAR(4) | sequence_no - Sequence Number monitor element |
| AUTH_ID | VARCHAR(128) | auth_id - Authorization ID monitor element |
| CLIENT_NNAME | VARCHAR(128) | client_nname - Configuration NNAME of Client monitor element |
| CLIENT_PRDID | VARCHAR(128) | client_prdid - Client Product/Version ID monitor element |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input Database Alias monitor element |
| CLIENT_DB_ALIAS | VARCHAR(128) | client_db_alias - Database Alias Used by Application monitor element |
| DB_NAME | VARCHAR(128) | db_name - Database Name monitor element |
| DB_PATH | VARCHAR(1024) | db_path - Database Path monitor element |
| EXECUTION_ID | VARCHAR(128) | execution_id - User Login ID monitor element |

SNAPSHOT_APPL_INFO

Table 145. Information returned by the SNAPSHOT_APPL_INFO table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|--------------|---|
| CORR_TOKEN | VARCHAR(128) | corr_token - DRDA Correlation Token monitor element |
| TPMON_CLIENT_USERID | VARCHAR(256) | tpmon_client_userid - TP Monitor Client User ID monitor element |
| TPMON_CLIENT_WKSTN | VARCHAR(256) | tpmon_client_wkstn - TP Monitor Client Workstation Name monitor element |
| TPMON_CLIENT_APP | VARCHAR(256) | tpmon_client_app - TP Monitor Client Application Name monitor element |
| TPMON_ACC_STR | VARCHAR(200) | tpmon_acc_str - TP Monitor Client Accounting String monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_BP

Note: This table function has been deprecated and replaced by the “SNAPBP administrative view and SNAP_GET_BP table function – Retrieve bufferpool logical group snapshot information” on page 341.

►►—SNAPSHOT_BP—(—dbname—,—dbpartitionnum—)—————►►

The schema is SYSPROC.

The SNAPSHOT_BP function returns information from a buffer pool snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 146. Information returned by the SNAPSHOT_BP table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer Pool Data Writes monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer Pool Index Writes monitor element |

Table 146. Information returned by the SNAPSHOT_BP table function (continued)

| Column name | Data type | Description or corresponding 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 |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer Pool Asynchronous Data Reads monitor element |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer Pool Asynchronous Data Writes monitor element |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer Pool Asynchronous Index Writes monitor element |
| 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 |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests monitor element |
| DIRECT_READS | BIGINT | direct_reads - Direct Reads From Database monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct Writes to Database 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 |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct Read Time monitor element |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct Write Time monitor element |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer Pool Asynchronous Index Reads monitor element |
| POOL_DATA_TO_ESTORE | BIGINT | pool_data_to_estore - Buffer Pool Data Pages to Extended Storage monitor element |
| POOL_INDEX_TO_ESTORE | BIGINT | pool_index_to_estore - Buffer Pool Index Pages to Extended Storage monitor element |
| POOL_INDEX_FROM_ESTORE | BIGINT | pool_index_from_estore - Buffer Pool Index Pages from Extended Storage monitor element |

Table 146. Information returned by the SNAPSHOT_BP table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|---------------|--|
| POOL_DATA_FROM_ESTORE | BIGINT | pool_data_from_estore - Buffer Pool Data Pages from Extended Storage monitor element |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread Prefetch Pages monitor element |
| FILES_CLOSED | BIGINT | files_closed - Database Files Closed monitor element |
| BP_NAME | VARCHAR(128) | bp_name - Buffer Pool Name monitor element |
| DB_NAME | VARCHAR(128) | db_name - Database Name monitor element |
| DB_PATH | VARCHAR(1024) | db_path - Database Path monitor element |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input Database Alias monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_CONTAINER

Note: This table function has been deprecated and replaced by the “SNAPCONTAINER administrative view and SNAP_GET_CONTAINER_V91 table function – Retrieve tablespace_container logical data group snapshot information” on page 351

▶▶—SNAPSHOT_CONTAINER—(—dbname—,—dbpartitionnum—)————▶▶

The schema is SYSPROC.

The SNAPSHOT_CONTAINER function returns container configuration information from a tablespace snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 147. Information returned by the SNAPSHOT_CONTAINER table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABLESPACE_ID | BIGINT | tablespace_id - Table Space Identification monitor element |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |
| CONTAINER_ID | BIGINT | container_id - Container Identification monitor element |
| CONTAINER_NAME | VARCHAR(256) | container_name - Container Name monitor element |
| CONTAINER_TYPE | SMALLINT | container_type - Container Type monitor element |
| TOTAL_PAGES | BIGINT | container_total_pages - Total Pages in Container monitor element |

Table 147. Information returned by the SNAPSHOT_CONTAINER table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------|-----------|--|
| USABLE_PAGES | BIGINT | container_usable_pages - Usable Pages in Container monitor element |
| ACCESSIBLE | BIGINT | container_accessible - Accessibility of Container monitor element |
| STRIPE_SET | BIGINT | container_stripe_set - Stripe Set monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_DATABASE

Note: This table function has been deprecated and replaced by the “SNAPDB administrative view and SNAP_GET_DB_V91 table function – Retrieve snapshot information from the dbase logical group” on page 356

▶▶—SNAPSHOT_DATABASE—(—dbname—,—dbpartitionnum—)————▶▶

The schema is SYSPROC.

The SNAPSHOT_DATABASE function returns information from a database snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 148. Information returned by the SNAPSHOT_DATABASE table function

| Column name | Data type | Description or corresponding monitor element |
|---------------------|-----------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| 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 |
| TOTAL_LOG_USED | BIGINT | total_log_used - Total Log Space Used monitor element |
| TOTAL_LOG_AVAILABLE | BIGINT | total_log_available - Total Log Available monitor element |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |

Table 148. Information returned by the SNAPSHOT_DATABASE 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 monitor element |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer Pool Data Writes monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |
| POOL_INDEX_P_READS | BIGINT | pool_index_p_reads - Buffer Pool Index Physical Reads monitor element |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer Pool Index 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 |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer Pool Asynchronous Index Reads monitor element |
| POOL_DATA_TO_ESTORE | BIGINT | pool_data_to_estore - Buffer Pool Data Pages to Extended Storage monitor element |
| POOL_INDEX_TO_ESTORE | BIGINT | pool_index_to_estore - Buffer Pool Index Pages to Extended Storage monitor element |
| POOL_INDEX_FROM_ESTORE | BIGINT | pool_index_from_estore - Buffer Pool Index Pages from Extended Storage monitor element |
| POOL_DATA_FROM_ESTORE | BIGINT | pool_data_from_estore - Buffer Pool Data Pages from Extended Storage monitor element |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer Pool Asynchronous Data Reads monitor element |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer Pool Asynchronous Data Writes monitor element |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer Pool Asynchronous Index Writes monitor element |
| 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 |

SNAPSHOT_DATABASE

Table 148. Information returned by the SNAPSHOT_DATABASE 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 monitor element |
| DIRECT_READS | BIGINT | direct_reads - Direct Reads From Database monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct Writes to Database 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 |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct Read Time monitor element |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct Write Time 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_LSN_GAP_CLNS | BIGINT | pool_lsn_gap_clns - Buffer Pool Log Space Cleaners Triggered monitor element |
| POOL_DRTY_PG_STEAL_CLNS | BIGINT | pool_drtly_pg_steal_clns - Buffer Pool Victim Page Cleaners Triggered monitor element |
| POOL_DRTY_PG_THRSH_CLNS | BIGINT | pool_drtly_pg_thrsh_clns - Buffer Pool Threshold Cleaners Triggered monitor element |
| LOCKS_HELD | BIGINT | locks_held - Locks Held monitor element |
| LOCK_WAITS | BIGINT | lock_waits - Lock Waits monitor element |
| LOCK_WAIT_TIME | BIGINT | lock_wait_time - Time Waited On Locks monitor element |
| LOCK_LIST_IN_USE | BIGINT | lock_list_in_use - Total Lock List Memory In Use monitor element |
| DEADLOCKS | BIGINT | deadlocks - Deadlocks Detected monitor element |
| LOCK_ESCALS | BIGINT | lock_escals - Number of Lock Escalations monitor element |
| X_LOCK_ESCALS | BIGINT | x_lock_escals - Exclusive Lock Escalations monitor element |
| LOCKS_WAITING | BIGINT | locks_waiting - Current Agents Waiting On Locks monitor element |
| SORT_HEAP_ALLOCATED | BIGINT | sort_heap_allocated - Total Sort Heap Allocated monitor element |

Table 148. Information returned by the SNAPSHOT_DATABASE table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|------------------|---|
| TOTAL_SORTS | BIGINT | total_sorts - Total Sorts monitor element |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total Sort Time monitor element |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort Overflows monitor element |
| ACTIVE_SORTS | BIGINT | active_sorts - Active Sorts monitor element |
| COMMIT_SQL_STMTS | BIGINT | commit_sql_stmts - Commit Statements Attempted monitor element |
| ROLLBACK_SQL_STMTS | BIGINT | rollback_sql_stmts - Rollback Statements Attempted 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 |
| DDL_SQL_STMTS | BIGINT | ddl_sql_stmts - Data Definition Language (DDL) SQL Statements monitor element |
| UID_SQL_STMTS | BIGINT | uid_sql_stmts - Update/Insert/Delete SQL Statements Executed monitor element |
| INT_AUTO_REBINDS | BIGINT | int_auto_rebinds - Internal Automatic Rebinds monitor element |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal Rows Updated monitor element |
| INT_COMMITS | BIGINT | int_commits - Internal Commits monitor element |
| INT_ROLLBACKS | BIGINT | int_rollbacks - Internal Rollbacks monitor element |
| INT_DEADLOCK_ROLLBACKS | BIGINT | int_deadlock_rollbacks - Internal Rollbacks Due To Deadlock monitor element |
| ROWS_DELETED | BIGINT | rows_deleted - Rows Deleted monitor element |

SNAPSHOT_DATABASE

Table 148. Information returned by the SNAPSHOT_DATABASE table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|-----------|---|
| ROWS_INSERTED | BIGINT | rows_inserted - Rows Inserted monitor element |
| ROWS_UPDATED | BIGINT | rows_updated - Rows Updated monitor element |
| ROWS_SELECTED | BIGINT | rows_selected - Rows Selected monitor element |
| BINDS_PRECOMPILES | BIGINT | binds_precompiles - Binds/Precompiles Attempted monitor element |
| TOTAL_CONS | BIGINT | total_cons - Connects Since Database Activation 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 |
| SEC_LOGS_ALLOCATED | BIGINT | sec_logs_allocated - Secondary Logs Allocated Currently monitor element |
| DB_STATUS | BIGINT | db_status - Status of Database monitor element |
| LOCK_TIMEOUTS | BIGINT | lock_timeouts - Number of Lock Timeouts monitor element |
| CONNECTIONS_TOP | BIGINT | connections_top - Maximum Number of Concurrent Connections monitor element |
| DB_HEAP_TOP | BIGINT | db_heap_top - Maximum Database Heap Allocated monitor element |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal Rows Inserted monitor element |
| LOG_READS | BIGINT | log_reads - Number of Log Pages Read monitor element |
| LOG_WRITES | BIGINT | log_writes - Number of Log Pages Written monitor element |
| PKG_CACHE_LOOKUPS | BIGINT | pkg_cache_lookups - Package Cache Lookups monitor element |
| PKG_CACHE_INSERTS | BIGINT | pkg_cache_inserts - Package Cache Inserts monitor element |
| CAT_CACHE_LOOKUPS | BIGINT | cat_cache_lookups - Catalog Cache Lookups monitor element |
| CAT_CACHE_INSERTS | BIGINT | cat_cache_inserts - Catalog Cache Inserts monitor element |
| CAT_CACHE_OVERFLOWS | BIGINT | cat_cache_overflows - Catalog Cache Overflows monitor element |

Table 148. Information returned by the SNAPSHOT_DATABASE table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|------------------|--|
| CAT_CACHE_HEAP_FULL | BIGINT | cat_cache_overflows - Catalog Cache Overflows monitor element |
| CATALOG_PARTITION | SMALLINT | catalog_node - Catalog Node Number monitor element |
| TOTAL_SEC_CONS | BIGINT | total_sec_cons - Secondary Connections 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 |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum Number of Coordinating Agents monitor element |
| PREFETCH_WAIT_TIME | BIGINT | prefetch_wait_time - Time Waited for Prefetch monitor element |
| APPL_SECTION_LOOKUPS | BIGINT | appl_section_lookups - Section Lookups monitor element |
| APPL_SECTION_INSERTS | BIGINT | appl_section_inserts - Section Inserts 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 |
| PKG_CACHE_NUM_OVERFLOWS | BIGINT | pkg_cache_num_overflows - Package Cache Overflows monitor element |
| PKG_CACHE_SIZE_TOP | BIGINT | pkg_cache_size_top - Package Cache High Water Mark monitor element |
| DB_CONN_TIME | TIMESTAMP | db_conn_time - Database Activation Timestamp monitor element |
| SQLM_ELM_LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |
| SQLM_ELM_LAST_BACKUP | TIMESTAMP | last_backup - Last Backup Timestamp monitor element |
| APPL_CON_TIME | TIMESTAMP | appl_con_time - Connection Request Start Timestamp monitor element |
| DB_LOCATION | INTEGER | db_location - Database Location monitor element |

SNAPSHOT_DATABASE

Table 148. Information returned by the SNAPSHOT_DATABASE table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|---------------|---|
| SERVER_PLATFORM | INTEGER | server_platform - Server Operating System monitor element |
| APPL_ID_OLDEST_XACT | BIGINT | appl_id_oldest_xact - Application with Oldest Transaction monitor element |
| CATALOG_PARTITION_NAME | VARCHAR(128) | catalog_node_name - Catalog Node Network Name monitor element |
| INPUT_DB_ALIAS | VARCHAR(128) | input_db_alias - Input Database Alias monitor element |
| DB_NAME | VARCHAR(128) | db_name - Database Name monitor element |
| DB_PATH | VARCHAR(1024) | db_path - Database Path monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_DBM

Note: This table function has been deprecated and replaced by the “SNAPDBM administrative view and SNAP_GET_DBM table function – Retrieve the dbm logical grouping snapshot information” on page 374.

►►—SNAPSHOT_DBM—(—*dbpartitionnum*—)—————►►

The schema is SYSPROC.

The SNAPSHOT_DBM function returns information from a snapshot of the DB2 database manager.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. If the null value is specified, -1 is set implicitly.

If the null value is specified, the snapshot will be taken only if a file has not previously been created by the SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 149. Information returned by the SNAPSHOT_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 monitor element |
| POST_THRESHOLD_SORTS | BIGINT | post_threshold_sorts - Post Threshold Sorts monitor element |
| PIPED_SORTS_REQUESTED | BIGINT | pipedsortsrequested - Piped Sorts Requested monitor element |
| PIPED_SORTS_ACCEPTED | BIGINT | pipedsortsaccepted - Piped Sorts Accepted monitor element |
| REM_CONS_IN | BIGINT | rem_cons_in - Remote Connections To Database Manager monitor element |
| 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 monitor element |
| 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 monitor element |

SNAPSHOT_DBM

Table 149. Information returned by the SNAPSHOT_DBM table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------------|-----------|--|
| AGENTS_REGISTERED | BIGINT | agents_registered - Agents Registered monitor element |
| AGENTS_WAITING_ON_TOKEN | BIGINT | agents_waiting_on_token - Agents Waiting for a Token monitor element |
| DB2_STATUS | BIGINT | db_status - Status of Database monitor element |
| AGENTS_REGISTERED_TOP | BIGINT | agents_registered_top - Maximum Number of Agents Registered monitor element |
| AGENTS_WAITING_TOP | BIGINT | agents_waiting_top - Maximum Number of Agents Waiting monitor element |
| COMM_PRIVATE_MEM | BIGINT | comm_private_mem - Committed Private Memory 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 |
| COORD_AGENTS_TOP | BIGINT | coord_agents_top - Maximum Number of Coordinating Agents monitor element |
| MAX_AGENT_OVERFLOW | BIGINT | max_agent_overflows - Maximum Agent Overflows 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 |
| POST_THRESHOLD_HASH_JOINS | BIGINT | post_threshold_hash_joins - Hash Join Threshold monitor element |
| INACTIVE_GW_AGENTS | BIGINT | idle_agents - Number of Idle Agents monitor element |

Table 149. Information returned by the SNAPSHOT_DBM table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|-----------|--|
| NUM_GW_CONN_SWITCHES | BIGINT | num_gw_conn_switches - Connection Switches monitor element |
| DB2START_TIME | TIMESTAMP | db2start_time - Start Database Manager Timestamp monitor element |
| LAST_RESET | TIMESTAMP | last_reset - Last Reset Timestamp monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_DYN_SQL

Note: This table function has been deprecated and replaced by the “SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information” on page 387

▶▶—SNAPSHOT_DYN_SQL—(—dbname—, —dbpartitionnum—)————▶▶

The schema is SYSPROC.

The SNAPSHOT_DYN_SQL function returns information from a dynamic SQL snapshot. It replaces the SQLCACHE_SNAPSHOT function, which is still available for compatibility reasons.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 150. Information returned by the SNAPSHOT_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. |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| ROWS_WRITTEN | BIGINT | rows_written - Rows Written monitor element |
| NUM_EXECUTIONS | BIGINT | num_executions - Statement Executions monitor element |
| NUM_COMPILATIONS | BIGINT | num_compilations - Statement Compilations monitor element |
| PREP_TIME_WORST | BIGINT | prep_time_worst - Statement Worst Preparation Time monitor element |
| PREP_TIME_BEST | BIGINT | prep_time_best - Statement Best Preparation Time monitor element |

Table 150. Information returned by the SNAPSHOT_DYN_SQL table function (continued)

| Column name | Data type | Description or corresponding 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 |
| STMT_SORTS | BIGINT | stmt_sorts - Statement Sorts monitor element |
| TOTAL_EXEC_TIME | BIGINT | total_exec_time - Elapsed Statement Execution Time monitor element |
| TOTAL_SYS_CPU_TIME | BIGINT | total_sys_cpu_time - Total System CPU for a Statement monitor element |
| TOTAL_USR_CPU_TIME | BIGINT | total_usr_cpu_time - Total User CPU for a Statement monitor element |
| STMT_TEXT | CLOB(16M) ¹ | stmt_text - SQL Dynamic Statement Text monitor element |

¹ STMT_TEXT is defined as CLOB(16M) to allow for future expansion only. Actual output of the statement text is truncated at 64K.

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_FCM

Note: This table function has been deprecated and replaced by the “SNAPFCM administrative view and SNAP_GET_FCM table function – Retrieve the fcm logical data group snapshot information” on page 392.

▶▶—SNAPSHOT_FCM—(—*dbpartitionnum*—)————▶▶

The schema is SYSPROC.

The SNAPSHOT_FCM function returns database manager level information regarding the fast communication manager (FCM).

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. If the null value is specified, -1 is set implicitly.

The function returns a table as shown below.

Table 151. Information returned by the SNAPSHOT_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 monitor element |
| BUFF_FREE_BOTTOM | BIGINT | buff_free_bottom - Minimum FCM Buffers Free monitor element |
| MA_FREE | BIGINT | ma_free - Message Anchors Currently Free monitor element |
| MA_FREE_BOTTOM | BIGINT | ma_free_bottom - Minimum Message Anchors monitor element |
| CE_FREE | BIGINT | ce_free - Connection Entries Currently Free monitor element |
| CE_FREE_BOTTOM | BIGINT | ce_free_bottom - Minimum Connection Entries monitor element |
| RB_FREE | BIGINT | rb_free - Request Blocks Currently Free monitor element |
| RB_FREE_BOTTOM | BIGINT | rb_free_bottom - Minimum Request Blocks monitor element |
| PARTITION_NUMBER | SMALLINT | node_number - Node Number monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_FCMNODE

Note: This table function has been deprecated and replaced by the “SNAPFCM_PART administrative view and SNAP_GET_FCM_PART table function – Retrieve the fcm_node logical data group snapshot information” on page 395.

▶▶—SNAPSHOT_FCMNODE—(—dbpartitionnum—)—————▶▶

The schema is SYSPROC.

The SNAPSHOT_FCMNODE function returns information from a snapshot of the fast communication manager in the database manager.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. If the null value is specified, -1 is set implicitly.

If the null value is specified, the snapshot will be taken only if a file has not previously been created by the SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 152. Information returned by the SNAPSHOT_FCMNODE 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 | BIGINT | connection_status - Connection Status monitor element |
| TOTAL_BUFFERS_SENT | BIGINT | total_buffers_sent - Total FCM Buffers Sent monitor element |
| TOTAL_BUFFERS_RCVD | BIGINT | total_buffers_rcvd - Total FCM Buffers Received monitor element |
| PARTITION_NUMBER | SMALLINT | node_number - Node Number monitor element |

Related tasks:

- “Capturing database system snapshot information to a file using the SNAP_WRITE_FILE stored procedure” in *System Monitor Guide and Reference*

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_FILEW

Note: This procedure has been deprecated and replaced by the “SNAP_WRITE_FILE procedure” on page 313.

►►—SNAPSHOT_FILEW—(—requestType—,—dbName—,—dbpartitionnum—)—————◄◄

The schema is SYSPROC.

The SNAPSHOT_FILEW procedure writes system snapshot data to a file located in the tmp subdirectory of the instance directory. To execute the SNAPSHOT_FILEW procedure, a user must have SYSADM, SYSCTRL, or SYSMANT authority. The saved snapshot can be read by users who do not have SYSADM, SYSCTRL, or SYSMANT authority by passing null values as the inputs to snapshot functions.

requestType

An input argument of type SMALLINT that specifies a valid snapshot request type, as defined in sqlmon.h.

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 procedure. Specify the null value to take the snapshot from the currently connected database.

dbpartitionnum

An input argument of type SMALLINT that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. If the null value is specified, -1 is set implicitly.

Example: Take a snapshot of database manager information by specifying a request type of 1 (which corresponds to SQLMA_DB2), and defaulting to the currently connected database and current database partition.

```
CALL SNAPSHOT_FILEW (1, CAST (NULL AS VARCHAR(128)), CAST (NULL AS SMALLINT))
```

This will result in snapshot data being written to /tmp/SQLMA_DB2.dat in the instance directory on UNIX operating systems or to \tmp\SQLMA_DB2.dat in the instance directory on a Windows operating system.

SNAPSHOT_LOCK

Note: This table function has been deprecated and replaced by the “SNAPLOCK administrative view and SNAP_GET_LOCK table function – Retrieve lock logical data group snapshot information” on page 403.

▶▶—SNAPSHOT_LOCK—(—dbname—,—dbpartitionnum—)—————▶▶

The schema is SYSPROC.

The SNAPSHOT_LOCK function returns information from a lock snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 153. Information returned by the SNAPSHOT_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) monitor element |
| TABLE_FILE_ID | BIGINT | table_file_id - Table File ID monitor element |
| LOCK_OBJECT_TYPE | BIGINT | lock_object_type - Lock Object Type Waited On monitor element |
| LOCK_MODE | BIGINT | lock_mode - Lock Mode monitor element |
| LOCK_STATUS | BIGINT | lock_status - Lock Status monitor element |
| LOCK_OBJECT_NAME | BIGINT | lock_object_name - Lock Object Name monitor element |

Table 153. Information returned by the SNAPSHOT_LOCK table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------|--------------|--|
| PARTITION_NUMBER | SMALLINT | node_number - Node Number monitor element |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock Escalation monitor element |
| TABLE_NAME | VARCHAR(128) | table_name - Table Name monitor element |
| TABLE_SCHEMA | VARCHAR(128) | table_schema - Table Schema Name monitor element |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_LOCKWAIT

Note: This table function has been deprecated and replaced by the “SNAPLOCKWAIT administrative view and SNAP_GET_LOCKWAIT table function – Retrieve lockwait logical data group snapshot information” on page 409.

▶▶—SNAPSHOT_LOCKWAIT—(—dbname—,—dbpartitionnum—)————▶▶

The schema is SYSPROC.

The SNAPSHOT_LOCKWAIT function returns lock waits information from an application snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 154. Information returned by the SNAPSHOT_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) monitor element |
| SUBSECTION_NUMBER | BIGINT | ss_number - Subsection Number monitor element |
| LOCK_MODE | BIGINT | lock_mode - Lock Mode monitor element |
| LOCK_OBJECT_TYPE | BIGINT | lock_object_type - Lock Object Type Waited On monitor element |
| AGENT_ID_HOLDING_LK | BIGINT | agent_id_holding_lock - Agent ID Holding Lock monitor element |

Table 154. Information returned by the SNAPSHOT_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 monitor element |
| LOCK_MODE_REQUESTED | BIGINT | lock_mode_requested - Lock Mode Requested monitor element |
| PARTITION_NUMBER | SMALLINT | node_number - Node Number monitor element |
| LOCK_ESCALATION | SMALLINT | lock_escalation - Lock Escalation monitor element |
| TABLE_NAME | VARCHAR(128) | table_name - Table Name monitor element |
| TABLE_SCHEMA | VARCHAR(128) | table_schema - Table Schema Name monitor element |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |
| APPL_ID_HOLDING_LK | VARCHAR(128) | appl_id_holding_lk - Application ID Holding Lock monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT QUIESCERS

Note: This table function has been deprecated and replaced by the “SNAPTbsp_QUIESCER administrative view and SNAP_GET_Tbsp_QUIESCER table function – Retrieve quiescer table space snapshot information” on page 452.

▶▶—SNAPSHOT QUIESCERS—(—dbname—, —dbpartitionnum—)————▶▶

The schema is SYSPROC.

The SNAPSHOT QUIESCERS function returns information about quiescers from a table space snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. If the null value is specified, -1 is set implicitly.

The function returns a table as shown below.

Table 155. Information returned by the SNAPSHOT QUIESCERS table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |
| QUIESCER_TBS_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 | BIGINT | quiescer_auth_id - Quiescer User Authorization Identification monitor element |
| QUIESCER_AGENT_ID | BIGINT | quiescer_agent_id - Quiescer Agent Identification monitor element |
| QUIESCER_STATE | BIGINT | quiescer_state - Quiescer State monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_RANGES

Note: This table function has been deprecated and replaced by the “SNAPTbsp_range administrative view and SNAP_GET_TBSP_RANGE table function – Retrieve range snapshot information” on page 456.

▶▶—SNAPSHOT_RANGES—(—dbname—,—dbpartitionnum—)—▶▶

The schema is SYSPROC.

The SNAPSHOT_RANGES function returns information from a range snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. If the null value is specified, -1 is set implicitly.

The function returns a table as shown below.

Table 156. Information returned by the SNAPSHOT_RANGES table function

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABLESPACE_ID | BIGINT | tablespace_id - Table Space Identification monitor element |
| TABLESPACE_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 |

Table 156. Information returned by the SNAPSHOT_RANGES table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|---------------------|-----------|--|
| RANGE_ADJUSTMENT | BIGINT | range_adjustment - Range Adjustment monitor element |
| RANGE_NUM_CONTAINER | BIGINT | range_num_containers - Number of Containers in Range monitor element |
| RANGE_CONTAINER_ID | BIGINT | range_container_id - Range Container monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_STATEMENT

Note: This table function has been deprecated and replaced by the “SNAPSTMT administrative view and SNAP_GET_STMT table function – Retrieve statement snapshot information” on page 415.

▶▶—SNAPSHOT_STATEMENT—(—dbname—,—dbpartitionnum—)————▶▶

The schema is SYSPROC.

The SNAPSHOT_STATEMENT function returns information about statements from an application snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 157. Information returned by the SNAPSHOT_STATEMENT 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) monitor element |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| ROWS_WRITTEN | BIGINT | rows_written - Rows Written 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 |
| STMT_TYPE | BIGINT | stmt_type - Statement Type monitor element |
| STMT_OPERATION | BIGINT | stmt_operation/operation - Statement Operation monitor element |

SNAPSHOT_STATEMENT

Table 157. Information returned by the SNAPSHOT_STATEMENT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|----------------------|-----------|--|
| SECTION_NUMBER | BIGINT | section_number - Section Number monitor element |
| QUERY_COST_ESTIMATE | BIGINT | query_cost_estimate - Query Cost Estimate monitor element |
| QUERY_CARD_ESTIMATE | BIGINT | query_card_estimate - Query Number of Rows Estimate monitor element |
| DEGREE_PARALLELISM | BIGINT | degree_parallelism - Degree of Parallelism monitor element |
| STMT_SORTS | BIGINT | stmt_sorts - Statement Sorts monitor element |
| TOTAL_SORT_TIME | BIGINT | total_sort_time - Total Sort Time monitor element |
| SORT_OVERFLOWS | BIGINT | sort_overflows - Sort Overflows monitor element |
| INT_ROWS_DELETED | BIGINT | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_UPDATED | BIGINT | int_rows_updated - Internal Rows Updated monitor element |
| INT_ROWS_INSERTED | BIGINT | int_rows_inserted - Internal Rows Inserted monitor element |
| FETCH_COUNT | BIGINT | fetch_count - Number of Successful Fetches monitor element |
| STMT_START | TIMESTAMP | stmt_start - Statement Operation Start Timestamp monitor element |
| STMT_STOP | TIMESTAMP | stmt_stop - Statement Operation Stop Timestamp monitor element |
| STMT_USR_CPU_TIME_S | BIGINT | stmt_usr_cpu_time - User CPU Time used by Statement monitor element |
| STMT_USR_CPU_TIME_MS | BIGINT | stmt_usr_cpu_time - User CPU Time used by Statement monitor element |
| STMT_SYS_CPU_TIME_S | BIGINT | stmt_sys_cpu_time - System CPU Time used by Statement monitor element |
| STMT_SYS_CPU_TIME_MS | BIGINT | stmt_sys_cpu_time - System CPU Time used by Statement monitor element |
| STMT_ELAPSED_TIME_S | BIGINT | stmt_elapsed_time - Most Recent Statement Elapsed Time monitor element |
| STMT_ELAPSED_TIME_MS | BIGINT | stmt_elapsed_time - Most Recent Statement Elapsed Time monitor element |
| BLOCKING_CURSOR | SMALLINT | blocking_cursor - Blocking Cursor monitor element |

SNAPSHOT_STATEMENT

Table 157. Information returned by the `SNAPSHOT_STATEMENT` table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|------------------------|--|
| STMT_PARTITION_NUMBER | SMALLINT | stmt_node_number - Statement Node monitor element |
| CURSOR_NAME | VARCHAR(128) | cursor_name - Cursor Name monitor element |
| CREATOR | VARCHAR(128) | creator - Application Creator monitor element |
| PACKAGE_NAME | VARCHAR(128) | package_name - Package Name monitor element |
| STMT_TEXT | CLOB(16M) ¹ | stmt_text - SQL Dynamic Statement Text monitor element |

¹ STMT_TEXT is defined as CLOB(16M) to allow for future expansion only. Actual output of the statement text is truncated at 64K.

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_SUBSECT

Note: This table function has been deprecated and replaced by the “SNAPSUBSECTION administrative view and SNAP_GET_SUBSECTION table function – Retrieve subsection logical monitor group snapshot information” on page 425.

►►—SNAPSHOT_SUBSECT—(—*dbname*—,—*dbpartitionnum*—)—————►►

The schema is SYSPROC.

The SNAPSHOT_SUBSECT function returns information about subsections of access plans from an application snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 158. Information returned by the SNAPSHOT_SUBSECT table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|------------------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| STMT_TEXT | CLOB(16M) ¹ | stmt_text - SQL Dynamic Statement Text monitor element |
| SS_EXEC_TIME | BIGINT | ss_exec_time - Subsection Execution Elapsed Time monitor element |
| TQ_TOT_SEND_SPILLS | BIGINT | tq_tot_send_spills - Total Number of Tablequeue Buffers Overflowed monitor element |
| TQ_CUR_SEND_SPILLS | BIGINT | tq_cur_send_spills - Current Number of Tablequeue Buffers Overflowed monitor element |
| TQ_MAX_SEND_SPILLS | BIGINT | tq_max_send_spills - Maximum Number of Tablequeue Buffers Overflows monitor element |
| TQ_ROWS_READ | BIGINT | tq_rows_read - Number of Rows Read from Tablequeues monitor element |

SNAPSHOT_SUBSECT

Table 158. Information returned by the SNAPSHOT_SUBSECT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------------|-----------|--|
| TQ_ROWS_WRITTEN | BIGINT | tq_rows_written - Number of Rows Written to Tablequeues monitor element |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| ROWS_WRITTEN | BIGINT | rows_written - Rows Written monitor element |
| SS_USR_CPU_TIME | BIGINT | ss_usr_cpu_time - User CPU Time used by Subsection monitor element |
| SS_SYS_CPU_TIME | BIGINT | ss_sys_cpu_time - System CPU Time used by Subsection monitor element |
| SS_NUMBER | INTEGER | ss_number - Subsection Number monitor element |
| SS_STATUS | INTEGER | ss_status - Subsection Status monitor element |
| SS_PARTITION_NUMBER | SMALLINT | ss_node_number - Subsection Node Number monitor element |
| TQ_PARTITION_WAITED_FOR | SMALLINT | tq_node_waited_for - Waited for Node on a Tablequeue monitor element |
| TQ_WAIT_FOR_ANY | INTEGER | tq_wait_for_any - Waiting for Any Node to Send on a Tablequeue monitor element |
| TQ_ID_WAITING_ON | INTEGER | tq_id_waiting_on - Waited on Node on a Tablequeue monitor element |

¹ STMT_TEXT is defined as CLOB(16M) to allow for future expansion only. Actual output of the statement text is truncated at 64K.

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_SWITCHES

Note: This table function has been deprecated and replaced by the “SNAPSWITCHES administrative view and SNAP_GET_SWITCHES table function – Retrieve database snapshot switch state information” on page 429.

►►—SNAPSHOT_SWITCHES—(—*dbpartitionnum*—)—————►►

The schema is SYSPROC.

The SNAPSHOT_SWITCHES function returns information about the database snapshot switch state.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. If the null value is specified, -1 is set implicitly.

The function returns a table as shown below.

Table 159. Information returned by the SNAPSHOT_SWITCHES table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|---|
| 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). |

SNAPSHOT_SWITCHES

Table 159. Information returned by the SNAPSHOT_SWITCHES table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------|-----------|---|
| 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). |
| 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. |
| PARTITION_NUMBER | SMALLINT | node_number - Node Number monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_TABLE

Note: This table function has been deprecated and replaced by the “SNAPTAB administrative view and SNAP_GET_TAB_V91 table function – Retrieve table logical data group snapshot information” on page 432

►►—SNAPSHOT_TABLE—(—dbname—,—dbpartitionnum—)—————►►

The schema is SYSPROC.

The SNAPSHOT_TABLE function returns activity information from a table snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 160. Information returned by the SNAPSHOT_TABLE table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| ROWS_WRITTEN | BIGINT | rows_written - Rows Written monitor element |
| ROWS_READ | BIGINT | rows_read - Rows Read monitor element |
| OVERFLOW_ACCESSES | BIGINT | overflow_accesses - Accesses to Overflowed Records monitor element |
| TABLE_FILE_ID | BIGINT | table_file_id - Table File ID monitor element |
| TABLE_TYPE | BIGINT | table_type - Table Type monitor element |
| PAGE_REORGS | BIGINT | page_reorgs - Page Reorganizations monitor element |

SNAPSHOT_TABLE

Table 160. Information returned by the SNAPSHOT_TABLE table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|--------------|--------------|--|
| TABLE_NAME | VARCHAR(128) | table_name - Table Name monitor element |
| TABLE_SCHEMA | VARCHAR(128) | table_schema - Table Schema Name monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_TBREORG

Note: This table function has been deprecated and replaced by the “SNAPTAB_REORG administrative view and SNAP_GET_TAB_REORG table function – Retrieve table reorganization snapshot information” on page 436.

►►—SNAPSHOT_TBREORG—(—*dbname*—,—*dbpartitionnum*—)—————►►

The schema is SYSPROC.

The SNAPSHOT_TBREORG function returns table reorganization information in the form of a result set. If no tables have been reorganized, 0 rows are returned. To obtain real-time snapshot information, the user must have SYSADM, SYSCTRL, or SYSMANT authority.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 161. Information returned by the SNAPSHOT_TBREORG table function

| Column name | Data type | Description or corresponding monitor element |
|-----------------------|--------------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABLE_NAME | VARCHAR(128) | table_name - Table Name monitor element |
| TABLE_SCHEMA | VARCHAR(128) | table_schema - Table Schema Name monitor element |
| PAGE_REORGS | BIGINT | page_reorgs - Page Reorganizations monitor element |
| REORG_PHASE | BIGINT | reorg_phase - Reorganize Phase monitor element |
| REORG_MAX_PHASE | INTEGER | reorg_max_phase - Maximum Reorganize Phase monitor element |
| REORG_CURRENT_COUNTER | BIGINT | reorg_current_counter - Reorganize Progress monitor element |

SNAPSHOT_TBREORG

Table 161. Information returned by the SNAPSHOT_TBREORG table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------|-----------|---|
| REORG_MAX_COUNTER | BIGINT | reorg_max_counter - Total Amount of Reorganization monitor element |
| REORG_TYPE | INTEGER | reorg_type - Table Reorganize Attributes monitor element |
| REORG_STATUS | BIGINT | reorg_status - Table Reorganize Status monitor element |
| REORG_COMPLETION | INTEGER | reorg_completion - Reorganization Completion Flag monitor element |
| REORG_START | TIMESTAMP | reorg_start - Table Reorganize Start Time monitor element |
| REORG_END | TIMESTAMP | reorg_end - Table Reorganize End Time monitor element |
| REORG_PHASE_START | TIMESTAMP | reorg_phase_start - Reorganize Phase Start Time monitor element |
| REORG_INDEX_ID | BIGINT | reorg_index_id - Index Used to Reorganize the Table monitor element |
| REORG_TBSPC_ID | BIGINT | reorg_tbspc_id - Table Space Where Table or Data partition is Reorganized monitor element |
| PARTITION_NUMBER | SMALLINT | node_number - Node Number monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_TBS

Note: This table function has been deprecated and replaced by the “SNAPTbsp administrative view and SNAP_GET_Tbsp_V91 table function – Retrieve tablespace logical data group snapshot information” on page 441

►►—SNAPSHOT_TBS—(—dbname—,—dbpartitionnum—)—————►►

The schema is SYSPROC.

The SNAPSHOT_TBS function returns activity information from a table space snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 162. Information returned by the SNAPSHOT_TBS table function

| Column name | Data type | Description or corresponding monitor element |
|------------------------|-----------|---|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| POOL_DATA_L_READS | BIGINT | pool_data_l_reads - Buffer Pool Data Logical Reads monitor element |
| POOL_DATA_P_READS | BIGINT | pool_data_p_reads - Buffer Pool Data Physical Reads monitor element |
| POOL_ASYNC_DATA_READS | BIGINT | pool_async_data_reads - Buffer Pool Asynchronous Data Reads monitor element |
| POOL_DATA_WRITES | BIGINT | pool_data_writes - Buffer Pool Data Writes monitor element |
| POOL_ASYNC_DATA_WRITES | BIGINT | pool_async_data_writes - Buffer Pool Asynchronous Data Writes monitor element |
| POOL_INDEX_L_READS | BIGINT | pool_index_l_reads - Buffer Pool Index Logical Reads monitor element |

Table 162. Information returned by the SNAPSHOT_TBS 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 monitor element |
| POOL_INDEX_WRITES | BIGINT | pool_index_writes - Buffer Pool Index Writes monitor element |
| POOL_ASYNC_INDEX_WRITES | BIGINT | pool_async_index_writes - Buffer Pool Asynchronous Index 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 |
| 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 |
| POOL_ASYNC_DATA_READ_REQS | BIGINT | pool_async_data_read_reqs - Buffer Pool Asynchronous Read Requests monitor element |
| DIRECT_READS | BIGINT | direct_reads - Direct Reads From Database monitor element |
| DIRECT_WRITES | BIGINT | direct_writes - Direct Writes to Database 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 |
| DIRECT_READ_TIME | BIGINT | direct_read_time - Direct Read Time monitor element |
| DIRECT_WRITE_TIME | BIGINT | direct_write_time - Direct Write Time monitor element |
| UNREAD_PREFETCH_PAGES | BIGINT | unread_prefetch_pages - Unread Prefetch Pages monitor element |
| POOL_ASYNC_INDEX_READS | BIGINT | pool_async_index_reads - Buffer Pool Asynchronous Index Reads monitor element |
| POOL_DATA_TO_ESTORE | BIGINT | pool_data_to_estore - Buffer Pool Data Pages to Extended Storage monitor element |
| POOL_INDEX_TO_ESTORE | BIGINT | pool_index_to_estore - Buffer Pool Index Pages to Extended Storage monitor element |

Table 162. Information returned by the SNAPSHOT_TBS table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------|--------------|--|
| POOL_INDEX_FROM_ESTORE | BIGINT | pool_index_from_estore - Buffer Pool Index Pages from Extended Storage monitor element |
| POOL_DATA_FROM_ESTORE | BIGINT | pool_data_from_estore - Buffer Pool Data Pages from Extended Storage monitor element |
| FILES_CLOSED | BIGINT | files_closed - Database Files Closed monitor element |
| TABLESPACE_NAME | VARCHAR(128) | tablespace_name - Table Space Name monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SNAPSHOT_TBS_CFG

Note: This table function has been deprecated and replaced by the “SNAPTbsp_Part administrative view and SNAP_Get_Tbsp_Part_V91 table function – Retrieve tablespace_nodeinfo logical data group snapshot information” on page 447

▶▶—SNAPSHOT_TBS_CFG—(—dbname—,—dbpartitionnum—)————▶▶

The schema is SYSPROC.

The SNAPSHOT_TBS_CFG function returns configuration information from a table space snapshot.

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.

dbpartitionnum

An input argument of type INTEGER that specifies a valid database partition number. Specify -1 for the current database partition, or -2 for all database partitions. 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 SNAPSHOT_FILEW stored procedure for the corresponding snapshot API request type.

The function returns a table as shown below.

Table 163. Information returned by the SNAPSHOT_TBS_CFG table function

| Column name | Data type | Description or corresponding monitor element |
|---------------------|---------------|--|
| SNAPSHOT_TIMESTAMP | TIMESTAMP | The date and time that the snapshot was taken. |
| TABLESPACE_ID | BIGINT | tablespace_id - Table Space Identification monitor element |
| TABLESPACE_NAME | VARCHAR (128) | tablespace_name - Table Space Name monitor element |
| TABLESPACE_TYPE | SMALLINT | tablespace_type - Table Space Type monitor element |
| TABLESPACE_STATE | BIGINT | tablespace_state - Table Space State monitor element |
| NUM QUIESCERS | BIGINT | tablespace_num_quiescers - Number of Quiescers monitor element |
| STATE_CHANGE_OBJ_ID | BIGINT | tablespace_state_change_object_id - State Change Object Identification monitor element |

Table 163. Information returned by the SNAPSHOT_TBS_CFG table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|------------------------------|-----------|---|
| STATE_CHANGE_TBS_ID | BIGINT | tablespace_state_change_ts_id - State Change Table Space Identification monitor element |
| MIN_RECOVERY_TIME | TIMESTAMP | tablespace_min_recovery_time - Minimum Recovery Time For Rollforward monitor element |
| TBS_CONTENTS_TYPE | SMALLINT | tablespace_content_type - Table Space Contents Type monitor element |
| BUFFERPOOL_ID | BIGINT | tablespace_cur_pool_id - Buffer Pool Currently Being Used monitor element |
| NEXT_BUFFERPOOL_ID | BIGINT | tablespace_next_pool_id - Buffer Pool That Will Be Used at Next Startup monitor element |
| PAGE_SIZE | BIGINT | tablespace_page_size - Table Space Page Size monitor element |
| EXTENT_SIZE | BIGINT | tablespace_extent_size - Table Space Extent Size monitor element |
| PREFETCH_SIZE | BIGINT | tablespace_prefetch_size - Table Space Prefetch Size monitor element |
| TOTAL_PAGES | BIGINT | tablespace_total_pages - Total Pages in Table Space monitor element |
| USABLE_PAGES | BIGINT | tablespace_usable_pages - Usable Pages in Table Space monitor element |
| USED_PAGES | BIGINT | tablespace_used_pages - Used Pages in Table Space monitor element |
| FREE_PAGES | BIGINT | tablespace_free_pages - Free Pages in Table Space monitor element |
| PENDING_FREE_PAGES | BIGINT | tablespace_pending_free_pages - Pending Free Pages in Table Space monitor element |
| HIGH_WATER_MARK | BIGINT | pool_watermark - Memory Pool Watermark monitor element |
| REBALANCER_MODE | BIGINT | tablespace_rebalancer_mode - Rebalancer Mode monitor element |
| REBALANCER_EXTENTS_REMAINING | BIGINT | tablespace_rebalancer_extents_remaining - Total Number of Extents to be Processed by the Rebalancer monitor element |
| REBALANCER_EXTENTS_PROCESSED | BIGINT | tablespace_rebalancer_extents_processed - Number of Extents the Rebalancer has Processed monitor element |
| REBALANCER_PRIORITY | BIGINT | tablespace_rebalancer_priority - Current Rebalancer Priority monitor element |
| REBALANCER_START_TIME | TIMESTAMP | tablespace_rebalancer_start_time - Rebalancer Start Time monitor element |
| REBALANCER_RESTART_TIME | TIMESTAMP | tablespace_rebalancer_restart_time - Rebalancer Restart Time monitor element |

SNAPSHOT_TBS_CFG

Table 163. Information returned by the SNAPSHOT_TBS_CFG table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------|-----------|---|
| LAST_EXTENT_MOVED | BIGINT | tablespace_rebalancer_last_extent_moved - Last Extent Moved by the Rebalancer monitor element |
| NUM_RANGES | BIGINT | tablespace_num_ranges - Number of Ranges in the Table Space Map monitor element |
| NUM_CONTAINERS | BIGINT | tablespace_num_containers - Number of Containers in Table Space monitor element |

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SQLCACHE_SNAPSHOT

Note: This table function has been deprecated and replaced by the “SNAPDYN_SQL administrative view and SNAP_GET_DYN_SQL_V91 table function – Retrieve dynsql logical group snapshot information” on page 387

►►—SQLCACHE_SNAPSHOT—(—)—————►►

The schema is SYSFUN.

The SQLCACHE_SNAPSHOT function returns the results of a snapshot of the DB2 dynamic SQL statement cache.

The function does not take any arguments. It returns a table, as shown below.

Table 164. Information returned by SQLCACHE_SNAPSHOT table function

| Column name | Data type | Description or corresponding monitor element |
|--------------------|-----------|--|
| NUM_EXECUTIONS | INTEGER | num_executions - Statement Executions monitor element |
| NUM_COMPILATIONS | INTEGER | num_compilations - Statement Compilations monitor element |
| PREP_TIME_WORST | INTEGER | prep_time_worst - Statement Worst Preparation Time monitor element |
| PREP_TIME_BEST | INTEGER | prep_time_best - Statement Best Preparation Time monitor element |
| INT_ROWS_DELETED | INTEGER | int_rows_deleted - Internal Rows Deleted monitor element |
| INT_ROWS_INSERTED | INTEGER | int_rows_inserted - Internal Rows Inserted monitor element |
| ROWS_READ | INTEGER | rows_read - Rows Read monitor element |
| INT_ROWS_UPDATED | INTEGER | int_rows_updated - Internal Rows Updated monitor element |
| ROWS_WRITE | INTEGER | rows_written - Rows Written monitor element |
| STMT_SORTS | INTEGER | stmt_sorts - Statement Sorts monitor element |
| TOTAL_EXEC_TIME_S | INTEGER | total_exec_time - Elapsed Statement Execution Time monitor element |
| TOTAL_EXEC_TIME_MS | INTEGER | total_exec_time - Elapsed Statement Execution Time monitor element |

SQLCACHE_SNAPSHOT

Table 164. Information returned by SQLCACHE_SNAPSHOT table function (continued)

| Column name | Data type | Description or corresponding monitor element |
|-------------------|------------------------|---|
| TOT_U_CPU_TIME_S | INTEGER | total_usr_cpu_time - Total User CPU for a Statement monitor element |
| TOT_U_CPU_TIME_MS | INTEGER | total_usr_cpu_time - Total User CPU for a Statement monitor element |
| TOT_S_CPU_TIME_S | INTEGER | total_sys_cpu_time - Total System CPU for a Statement monitor element |
| TOT_S_CPU_TIME_MS | INTEGER | total_sys_cpu_time - Total System CPU for a Statement monitor element |
| DB_NAME | VARCHAR(128) | db_name - Database Name monitor element |
| STMT_TEXT | CLOB(16M) ¹ | stmt_text - SQL Dynamic Statement Text monitor element |

¹ STMT_TEXT is defined as CLOB(16M) to allow for future expansion only. Actual output of the statement text is truncated at 64K.

Related reference:

- “Snapshot monitor logical data groups and monitor elements” in *System Monitor Guide and Reference*

SYSINSTALLROUTINES

Note: This procedure has been deprecated. The procedure was used to create new procedures and functions in DB2 UDB for Linux, UNIX, and Windows Version 8.

▶▶—SYSINSTALLROUTINES—(—)—▶▶

The schema is SYSPROC.

SYSINSTALLROUTINES

Appendix A. DB2 Database technical information

Overview of the DB2 technical information

DB2 technical information is available through the following tools and methods:

- DB2 Information Center
 - Topics
 - Help for DB2 tools
 - Sample programs
 - Tutorials
- DB2 books
 - PDF files (downloadable)
 - PDF files (from the DB2 PDF CD)
 - printed books
- Command line help
 - Command help
 - Message help
- Sample programs

IBM® periodically makes documentation updates available. If you access the online version on the DB2 Information Center at ibm.com®, you do not need to install documentation updates because this version is kept up-to-date by IBM. If you have installed the DB2 Information Center, it is recommended that you install the documentation updates. Documentation updates allow you to update the information that you installed from the *DB2 Information Center CD* or downloaded from Passport Advantage as new information becomes available.

Note: The DB2 Information Center topics are updated more frequently than either the PDF or the hard-copy books. To get the most current information, install the documentation updates as they become available, or refer to the DB2 Information Center at ibm.com.

You can access additional DB2 technical information such as technotes, white papers, and Redbooks™ online at ibm.com. Access the DB2 Information Management software library site at <http://www.ibm.com/software/data/sw-library/>.

Documentation feedback

We value your feedback on the DB2 documentation. If you have suggestions for how we can improve the DB2 documentation, send an e-mail to db2docs@ca.ibm.com. The DB2 documentation team reads all of your feedback, but cannot respond to you directly. Provide specific examples wherever possible so that we can better understand your concerns. If you are providing feedback on a specific topic or help file, include the topic title and URL.

Do not use this e-mail address to contact DB2 Customer Support. If you have a DB2 technical issue that the documentation does not resolve, contact your local IBM service center for assistance.

Related concepts:

- “Features of the DB2 Information Center” in *Online DB2 Information Center*
- “Sample files” in *Samples Topics*

Related tasks:

- “Invoking command help from the command line processor” in *Command Reference*
- “Invoking message help from the command line processor” in *Command Reference*
- “Updating the DB2 Information Center installed on your computer or intranet server” on page 655

Related reference:

- “DB2 technical library in hardcopy or PDF format” on page 650

DB2 technical library in hardcopy or PDF format

The following tables describe the DB2 library available from the IBM Publications Center at www.ibm.com/shop/publications/order. DB2 Version 9 manuals in PDF format can be downloaded from www.ibm.com/software/data/db2/udb/support/manualsv9.html.

Although the tables identify books available in print, the books might not be available in your country or region.

The information in these books is fundamental to all DB2 users; you will find this information useful whether you are a programmer, a database administrator, or someone who works with DB2 Connect or other DB2 products.

Table 165. DB2 technical information

| Name | Form Number | Available in print |
|--|-------------|--------------------|
| <i>Administration Guide: Implementation</i> | SC10-4221 | Yes |
| <i>Administration Guide: Planning</i> | SC10-4223 | Yes |
| <i>Administrative API Reference</i> | SC10-4231 | Yes |
| <i>Administrative SQL Routines and Views</i> | SC10-4293 | No |
| <i>Call Level Interface Guide and Reference, Volume 1</i> | SC10-4224 | Yes |
| <i>Call Level Interface Guide and Reference, Volume 2</i> | SC10-4225 | Yes |
| <i>Command Reference</i> | SC10-4226 | No |
| <i>Data Movement Utilities Guide and Reference</i> | SC10-4227 | Yes |
| <i>Data Recovery and High Availability Guide and Reference</i> | SC10-4228 | Yes |
| <i>Developing ADO.NET and OLE DB Applications</i> | SC10-4230 | Yes |
| <i>Developing Embedded SQL Applications</i> | SC10-4232 | Yes |

Table 165. DB2 technical information (continued)

| Name | Form Number | Available in print |
|--|-------------|--------------------|
| <i>Developing SQL and External Routines</i> | SC10-4373 | No |
| <i>Developing Java Applications</i> | SC10-4233 | Yes |
| <i>Developing Perl and PHP Applications</i> | SC10-4234 | No |
| <i>Getting Started with Database Application Development</i> | SC10-4252 | Yes |
| <i>Getting started with DB2 installation and administration on Linux and Windows</i> | GC10-4247 | Yes |
| <i>Message Reference Volume 1</i> | SC10-4238 | No |
| <i>Message Reference Volume 2</i> | SC10-4239 | No |
| <i>Migration Guide</i> | GC10-4237 | Yes |
| <i>Net Search Extender Administration and User's Guide</i> Note: HTML for this document is not installed from the HTML documentation CD. | SH12-6842 | Yes |
| <i>Performance Guide</i> | SC10-4222 | Yes |
| <i>Query Patroller Administration and User's Guide</i> | GC10-4241 | Yes |
| <i>Quick Beginnings for DB2 Clients</i> | GC10-4242 | No |
| <i>Quick Beginnings for DB2 Servers</i> | GC10-4246 | Yes |
| <i>Spatial Extender and Geodetic Data Management Feature User's Guide and Reference</i> | SC18-9749 | Yes |
| <i>SQL Guide</i> | SC10-4248 | Yes |
| <i>SQL Reference, Volume 1</i> | SC10-4249 | Yes |
| <i>SQL Reference, Volume 2</i> | SC10-4250 | Yes |
| <i>System Monitor Guide and Reference</i> | SC10-4251 | Yes |
| <i>Troubleshooting Guide</i> | GC10-4240 | No |
| <i>Visual Explain Tutorial</i> | SC10-4319 | No |
| <i>What's New</i> | SC10-4253 | Yes |
| <i>XML Extender Administration and Programming</i> | SC18-9750 | Yes |
| <i>XML Guide</i> | SC10-4254 | Yes |
| <i>XQuery Reference</i> | SC18-9796 | Yes |

Table 166. DB2 Connect-specific technical information

| Name | Form Number | Available in print |
|---------------------------------|-------------|--------------------|
| <i>DB2 Connect User's Guide</i> | SC10-4229 | Yes |

Table 166. DB2 Connect-specific technical information (continued)

| Name | Form Number | Available in print |
|---|-------------|--------------------|
| Quick Beginnings for DB2 Connect Personal Edition | GC10-4244 | Yes |
| Quick Beginnings for DB2 Connect Servers | GC10-4243 | Yes |

Table 167. WebSphere® Information Integration technical information

| Name | Form Number | Available in print |
|--|-------------|--------------------|
| WebSphere Information Integration: Administration Guide for Federated Systems | SC19-1020 | Yes |
| WebSphere Information Integration: ASNCLP Program Reference for Replication and Event Publishing | SC19-1018 | Yes |
| WebSphere Information Integration: Configuration Guide for Federated Data Sources | SC19-1034 | No |
| WebSphere Information Integration: SQL Replication Guide and Reference | SC19-1030 | Yes |

Note: The DB2 Release Notes provide additional information specific to your product's release and fix pack level. For more information, see the related links.

Related concepts:

- "Overview of the DB2 technical information" on page 649
- "About the Release Notes" in *Release notes*

Related tasks:

- "Ordering printed DB2 books" on page 652

Ordering printed DB2 books

If you require printed DB2 books, you can buy them online in many but not all countries or regions. You can always order printed DB2 books from your local IBM representative. Keep in mind that some softcopy books on the *DB2 PDF Documentation CD* are unavailable in print. For example, neither volume of the *DB2 Message Reference* is available as a printed book.

Printed versions of many of the DB2 books available on the DB2 PDF Documentation CD can be ordered for a fee from IBM. Depending on where you are placing your order from, you may be able to order books online, from the IBM Publications Center. If online ordering is not available in your country or region, you can always order printed DB2 books from your local IBM representative. Note that not all books on the DB2 PDF Documentation CD are available in print.

Note: The most up-to-date and complete DB2 documentation is maintained in the DB2 Information Center at <http://publib.boulder.ibm.com/infocenter/db2help/>.

Procedure:

To order printed DB2 books:

- To find out whether you can order printed DB2 books online in your country or region, check the IBM Publications Center at <http://www.ibm.com/shop/publications/order>. You must select a country, region, or language to access publication ordering information and then follow the ordering instructions for your location.
- To order printed DB2 books from your local IBM representative:
 - Locate the contact information for your local representative from one of the following Web sites:
 - The IBM directory of world wide contacts at www.ibm.com/planetwide
 - The IBM Publications Web site at <http://www.ibm.com/shop/publications/order>. You will need to select your country, region, or language to the access appropriate publications home page for your location. From this page, follow the "About this site" link.
 - When you call, specify that you want to order a DB2 publication.
 - Provide your representative with the titles and form numbers of the books that you want to order.

Related concepts:

- "Overview of the DB2 technical information" on page 649

Related reference:

- "DB2 technical library in hardcopy or PDF format" on page 650

Displaying SQL state help from the command line processor

DB2 returns an SQLSTATE value for conditions that could be the result of an SQL statement. SQLSTATE help explains the meanings of SQL states and SQL state class codes.

Procedure:

To invoke SQL state help, open the command line processor and enter:

```
? sqlstate or ? class code
```

where *sqlstate* represents a valid five-digit SQL state and *class code* represents the first two digits of the SQL state.

For example, ? 08003 displays help for the 08003 SQL state, and ? 08 displays help for the 08 class code.

Related tasks:

- "Invoking command help from the command line processor" in *Command Reference*
- "Invoking message help from the command line processor" in *Command Reference*

Accessing different versions of the DB2 Information Center

For DB2 Version 9 topics, the DB2 Information Center URL is <http://publib.boulder.ibm.com/infocenter/db2luw/v9/>.

For DB2 Version 8 topics, go to the Version 8 Information Center URL at: <http://publib.boulder.ibm.com/infocenter/db2luw/v8/>.

Related tasks:

- “Setting up access to DB2 contextual help and documentation” in *Administration Guide: Implementation*

Displaying topics in your preferred language in the DB2 Information Center

The DB2 Information Center attempts to display topics in the language specified in your browser preferences. If a topic has not been translated into your preferred language, the DB2 Information Center displays the topic in English.

Procedure:

To display topics in your preferred language in the Internet Explorer browser:

1. In Internet Explorer, click the **Tools** → **Internet Options** → **Languages...** button. The Language Preferences window opens.
2. Ensure your preferred language is specified as the first entry in the list of languages.
 - To add a new language to the list, click the **Add...** button.

Note: Adding a language does not guarantee that the computer has the fonts required to display the topics in the preferred language.

- To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.

To display topics in your preferred language in a Firefox or Mozilla browser:

1. Select the **Tools** → **Options** → **Languages** button. The Languages panel is displayed in the Preferences window.
2. Ensure your preferred language is specified as the first entry in the list of languages.
 - To add a new language to the list, click the **Add...** button to select a language from the Add Languages window.
 - To move a language to the top of the list, select the language and click the **Move Up** button until the language is first in the list of languages.
3. Clear the browser cache and then refresh the page to display the DB2 Information Center in your preferred language.

On some browser and operating system combinations, you might have to also change the regional settings of your operating system to the locale and language of your choice.

Related concepts:

- “Overview of the DB2 technical information” on page 649

Updating the DB2 Information Center installed on your computer or intranet server

If you have a locally-installed DB2 Information Center, updated topics can be available for download. The 'Last updated' value found at the bottom of most topics indicates the current level for that topic.

To determine if there is an update available for the entire DB2 Information Center, look for the 'Last updated' value on the Information Center home page. Compare the value in your locally installed home page to the date of the most recent downloadable update at <http://www.ibm.com/software/data/db2/udb/support/icupdate.html>. You can then update your locally-installed Information Center if a more recent downloadable update is available.

Updating your locally-installed DB2 Information Center requires that you:

1. Stop the DB2 Information Center on your computer, and restart the Information Center in stand-alone mode. Running the Information Center in stand-alone mode prevents other users on your network from accessing the Information Center, and allows you to download and apply updates.
2. Use the Update feature to determine if update packages are available from IBM.

Note: Updates are also available on CD. For details on how to configure your Information Center to install updates from CD, see the related links. If update packages are available, use the Update feature to download the packages. (The Update feature is only available in stand-alone mode.)

3. Stop the stand-alone Information Center, and restart the DB2 Information Center service on your computer.

Procedure:

To update the DB2 Information Center installed on your computer or intranet server:

1. Stop the DB2 Information Center service.
 - On Windows, click **Start** → **Control Panel** → **Administrative Tools** → **Services**. Then right-click on **DB2 Information Center** service and select **Stop**.
 - On Linux®, enter the following command:
`/etc/init.d/db2icdv9 stop`
2. Start the Information Center in stand-alone mode.
 - On Windows:
 - a. Open a command window.
 - b. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the C:\Program Files\IBM\DB2 Information Center\Version 9 directory.
 - c. Run the help_start.bat file using the fully qualified path for the DB2 Information Center:
`<DB2 Information Center dir>\doc\bin\help_start.bat`
 - On Linux:

- a. Navigate to the path where the Information Center is installed. By default, the DB2 Information Center is installed in the /opt/ibm/db2ic/V9 directory.
- b. Run the help_start script using the fully qualified path for the DB2 Information Center:

```
<DB2 Information Center dir>/doc/bin/help_start
```

The systems default Web browser launches to display the stand-alone Information Center.

3. Click the Update button (🔄). On the right hand panel of the Information Center, click **Find Updates**. A list of updates for existing documentation displays.
4. To initiate the download process, check the selections you want to download, then click **Install Updates**.
5. After the download and installation process has completed, click **Finish**.
6. Stop the stand-alone Information Center.
 - On Windows, run the help_end.bat file using the fully qualified path for the DB2 Information Center:

```
<DB2 Information Center dir>\doc\bin\help_end.bat
```

Note: The help_end batch file contains the commands required to safely terminate the processes that were started with the help_start batch file. Do not use Ctrl-C or any other method to terminate help_start.bat.
 - On Linux, run the help_end script using the fully qualified path for the DB2 Information Center:

```
<DB2 Information Center dir>/doc/bin/help_end
```

Note: The help_end script contains the commands required to safely terminate the processes that were started with the help_start script. Do not use any other method to terminate the help_start script.
7. Restart the DB2 Information Center service.
 - On Windows, click **Start** → **Control Panel** → **Administrative Tools** → **Services**. Then right-click on **DB2 Information Center** service and select **Start**.
 - On Linux, enter the following command:

```
/etc/init.d/db2icdv9 start
```

The updated DB2 Information Center displays the new and updated topics.

Related concepts:

- “DB2 Information Center installation options” in *Quick Beginnings for DB2 Servers*

Related tasks:

- “Installing the DB2 Information Center using the DB2 Setup wizard (Linux)” in *Quick Beginnings for DB2 Servers*
- “Installing the DB2 Information Center using the DB2 Setup wizard (Windows)” in *Quick Beginnings for DB2 Servers*

DB2 tutorials

The DB2 tutorials help you learn about various aspects of DB2 products. Lessons provide step-by-step instructions.

Before you begin:

You can view the XHTML version of the tutorial from the Information Center at <http://publib.boulder.ibm.com/infocenter/db2help/>.

Some lessons use sample data or code. See the tutorial for a description of any prerequisites for its specific tasks.

DB2 tutorials:

To view the tutorial, click on the title.

Native XML data store

Set up a DB2 database to store XML data and to perform basic operations with the native XML data store.

Visual Explain Tutorial

Analyze, optimize, and tune SQL statements for better performance using Visual Explain.

Related concepts:

- “Visual Explain overview” in *Administration Guide: Implementation*

DB2 troubleshooting information

A wide variety of troubleshooting and problem determination information is available to assist you in using DB2 products.

DB2 documentation

Troubleshooting information can be found in the DB2 Troubleshooting Guide or the Support and Troubleshooting section of the DB2 Information Center. There you will find information on how to isolate and identify problems using DB2 diagnostic tools and utilities, solutions to some of the most common problems, and other advice on how to solve problems you might encounter with your DB2 products.

DB2 Technical Support Web site

Refer to the DB2 Technical Support Web site if you are experiencing problems and want help finding possible causes and solutions. The Technical Support site has links to the latest DB2 publications, TechNotes, Authorized Program Analysis Reports (APARs or bug fixes), fix packs, and other resources. You can search through this knowledge base to find possible solutions to your problems.

Access the DB2 Technical Support Web site at <http://www.ibm.com/software/data/db2/udb/support.html>

Related concepts:

- “Introduction to problem determination” in *Troubleshooting Guide*
- “Overview of the DB2 technical information” on page 649

Terms and Conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal use: You may reproduce these Publications for your personal, non commercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these Publications, or any portion thereof, without the express consent of IBM.

Commercial use: You may reproduce, distribute and display these Publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these Publications, or reproduce, distribute or display these Publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the Publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Appendix B. Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country/region or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country/region where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product, and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs, in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (*your company name*) (*year*). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. *_enter the year or years_*. All rights reserved.

Trademarks

Company, product, or service names identified in the documents of the DB2 Version 9 documentation library may be trademarks or service marks of International Business Machines Corporation or other companies. Information on the trademarks of IBM Corporation in the United States, other countries, or both is located at <http://www.ibm.com/legal/copytrade.shtml>.

The following terms are trademarks or registered trademarks of other companies and have been used in at least one of the documents in the DB2 documentation library:

Microsoft[®], Windows, Windows NT[®], and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel[®], Itanium[®], Pentium[®], and Xeon[®] are trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Index

A

ADD CONTACT command
 using ADMIN_CMD 44
ADD CONTACTGROUP command
 using ADMIN_CMD 46
ADMIN_CMD procedure 38
 removing messages 43
 retrieving messages 41
 supported commands
 ADD CONTACT 44
 ADD CONTACTGROUP 46
 AUTOCONFIGURE 48
 BACKUP DATABASE 53
 DESCRIBE 58
 DROP CONTACT 68
 DROP CONTACTGROUP 69
 EXPORT 70
 FORCE APPLICATION 76
 GET STMM TUNING
 DBPARTITIONNUM 78
 IMPORT 80
 INITIALIZE TAPE 94
 LOAD 96
 PRUNE HISTORY/LOGFILE 115
 QUIESCE DATABASE 117
 QUIESCE TABLESPACES FOR
 TABLE 119
 REDISTRIBUTE DATABASE
 PARTITION GROUP 122
 REORG INDEXES/TABLE 126
 RESET ALERT
 CONFIGURATION 136
 RESET DATABASE
 CONFIGURATION 139
 RESET DATABASE MANAGER
 CONFIGURATION 141
 REWIND TAPE 143
 RUNSTATS 144
 SET TAPE POSITION 156
 UNQUIESCE DATABASE 157
 UPDATE ALERT
 CONFIGURATION 159
 UPDATE CONTACT 164
 UPDATE CONTACTGROUP 166
 UPDATE DATABASE
 CONFIGURATION 168
 UPDATE DATABASE MANAGER
 CONFIGURATION 171
 UPDATE HEALTH
 NOTIFICATION CONTACT
 LIST 174
 UPDATE HISTORY 176
 UPDATE STMM TUNING
 DBPARTITIONNUM 178
ADMIN_COPY_SCHEMA
 procedure 498
ADMIN_DROP_SCHEMA
 procedure 503
ADMIN_GET_MSGS table function 41
ADMIN_GET_TAB_INFO table
 function 506

ADMIN_REMOVE_MSGS procedure 43
administrative SQL routines
 supported 8
administrative views 2
 ADMINTABINFO 506
 APPL_PERFORMANCE 286
 APPLICATIONS 280
 authorization 6
 AUTHORIZATIONIDS 275
 BP_HITRATIO 288
 BP_READ_IO 290
 BP_WRITE_IO 292
 CONTACTGROUPS 523
 CONTACTS 525
 CONTAINER_UTILIZATION 294
 DB_HISTORY 527
 DBCFG 182
 DBMCFG 184
 DBPATHS 532
 ENV_INST_INFO 189
 ENV_PROD_INFO 191
 ENV_SYS_INFO 193
 LOCKS_HELD 297
 LOCKWAIT 301
 LOG_UTILIZATION 306
 LONG_RUNNING_SQL 308
 NOTIFICATIONLIST 542
 OBJECTOWNERS 276
 PDLOGMSG_LAST24HOURS 543
 PRIVILEGES 278
 QUERY_PREP_COST 311
 REG_VARIABLES 187
 SNAPAGENT 315
 SNAPAGENT_MEMORY_POOL 319
 SNAPAPPL 324
 SNAPAPPL_INFO 334
 SNAPBP 341
 SNAPBP_PART 347
 SNAPCONTAINER 351
 SNAPDB 356
 SNAPDB_MEMORY_POOL 369
 SNAPDBM 374
 SNAPDBM_MEMORY_POOL 379
 SNAPDETAILLOG 383
 SNAPDYN_SQL 387
 SNAPFCM 392
 SNAPFCM_PART 395
 SNAPHADR 398
 SNAPLOCK 403
 SNAPLOCKWAIT 409
 SNAPSTMT 415
 SNAPSTORAGE_PATHS 421
 SNAPSUBSECTION 425
 SNAPSWITCHES 429
 SNAPTAB 432
 SNAPTAB_REORG 436
 SNAPTbsp 441
 SNAPTbsp_QUIESCE 452
 SNAPTbsp_RANGE 456
 SNAPTbspPART 447
 SNAPUTIL 460

administrative views (*continued*)
 SNAPUTIL_PROGRESS 464
 supported 8
 TBSP_UTILIZATION 467
 TOP_DYNAMIC_SQL 472
 versus table functions 3
ADMINTABINFO administrative
 view 506
ALTOBJ procedure 516
AM_BASE_RPT_RECOMS table
 function 18
AM_BASE_RPTS table function 20
AM_DROP_TASK procedure 22
AM_GET_LOCK_CHN_TB procedure 23
AM_GET_LOCK_CHNS procedure 25
AM_GET_LOCK_RPT procedure 26
AM_GET_RPT procedure 34
AM_SAVE_TASK procedure 36
ANALYZE_LOG_SPACE procedure 480
APPL_PERFORMANCE administrative
 view 286
APPLICATION_ID scalar function 519
APPLICATIONS administrative
 view 280
AUTH_LIST_GROUPS_FOR_AUTHID
 table function 273
AUTHORIZATIONIDS administrative
 view 275
 authorizations
 for administrative views 6
 retrieving authorization IDs 275
 retrieving group membership 273
AUTOCONFIGURE command
 using ADMIN_CMD 48

B

BACKUP DATABASE command
 using ADMIN_CMD 53
BP_HITRATIO administrative view 288
BP_READ_IO administrative view 290
BP_WRITE_IO administrative view 292

C

CAPTURE_STORAGE_MGMT_INFO
 procedure 493
commands
 ADD CONTACT 44
 ADD CONTACTGROUP 46
 AUTOCONFIGURE 48
 BACKUP DATABASE 53
 calling from a procedure 38
 DESCRIBE 58
 DROP CONTACT 68
 DROP CONTACTGROUP 69
 EXPORT 70
 FORCE APPLICATION 76
 GET STMM TUNING
 DBPARTITIONNUM 78

commands (*continued*)

- IMPORT 80
- INITIALIZE TAPE 94
- LOAD 96
- PRUNE HISTORY/LOGFILE 115
- QUIESCE DATABASE 117
- QUIESCE TABLESPACES FOR TABLE 119
- REDISTRIBUTE DATABASE PARTITION GROUP 122
- REORG INDEXES/TABLE 126
- RESET ALERT CONFIGURATION 136
- RESET DATABASE CONFIGURATION 139
- RESET DATABASE MANAGER CONFIGURATION 141
- REWIND TAPE 143
- RUNSTATS 144
- SET TAPE POSITION 156
- UNQUIESCE DATABASE 157
- UPDATE ALERT CONFIGURATION 159
- UPDATE CONTACT 164
- UPDATE CONTACTGROUP 166
- UPDATE DATABASE CONFIGURATION 168
- UPDATE DATABASE MANAGER CONFIGURATION 171
- UPDATE HEALTH NOTIFICATION CONTACT LIST 174
- UPDATE HISTORY 176
- UPDATE STMM TUNING DBPARTITIONNUM 178
- COMPILATION_ENV table function 520
- contact lists
 - retrieving contact groups lists 523
 - retrieving contacts 525
- CONTACTGROUPS administrative view 523
- contacting IBM 665
- contacts
 - retrieving contact groups 523
 - retrieving contact lists 525
- CONTACTS administrative view 525
- CONTAINER_UTILIZATION administrative view 294
- copying
 - schemas and objects 498
- CREATE_STORAGEMGMT_TABLES procedure 495

D

- database configuration
 - retrieving parameters 182
- database manager configuration
 - retrieving parameters 184
- database paths
 - retrieving 532
- DB_HISTORY administrative view 527
- DB_PARTITIONS table function 180
- DB2 Information Center
 - updating 655
 - versions 654
 - viewing in different languages 654
- DBCFG administrative view 182

- DBMCFG administrative view 184
- DBPATHS administrative view 532
- deprecated
 - procedures 563, 619, 647
 - table functions 565, 566, 568, 576, 579, 580, 582, 586, 589, 590, 596, 599, 602, 604, 611, 614, 616, 618, 620, 622, 624, 626, 628, 631, 633, 635, 637, 639, 642, 645
- deprecated functionality
 - SQL administrative routines 559
- DESCRIBE command
 - using ADMIN_CMD 58
- documentation 649, 650
- terms and conditions of use 658
- DROP CONTACT command
 - using ADMIN_CMD 68
- DROP CONTACTGROUP command
 - using ADMIN_CMD 69
- DROP_STORAGEMGMT_TABLES procedure 497
- dropping
 - schemas schemas and objects 503

E

- ENV_INST_INFO administrative view 189
- ENV_PROD_INFO administrative view 191
- ENV_SYS_INFO administrative view 193
- error messages
 - retrieving information 555
- EXPLAIN_GET_MSGS table function 536
- EXPORT command
 - using ADMIN_CMD 70

F

- FORCE APPLICATION command
 - using ADMIN_CMD 76
- functions
 - scalar
 - APPLICATION_ID 519
 - GET_ROUTINE_OPTS 474
 - MQPUBLISH 244
 - MQREAD 247
 - MQREADCLOB 255
 - MQRECEIVE 257
 - MQRECEIVECLOB 265
 - MQSEND 267
 - MQSUBSCRIBE 269
 - MQUNSUBSCRIBE 271
 - SQLERRM 555
 - supported 8
 - table functions 2
 - ADMIN_GET_MSGS 41
 - ADMIN_GET_TAB_INFO 506
 - AM_BASE_RPT_RECOMS 18
 - AM_BASE_RPTS 20
 - AUTH_LIST_GROUPS_FOR_AUTHID 273
 - COMPILATION_ENV 520
 - DB_PARTITIONS 180
 - deprecated 559

functions (*continued*)

- table functions (*continued*)
 - EXPLAIN_GET_MSGS 536
 - GET_DB_CONFIG 563
 - GET_DBM_CONFIG 565
 - HEALTH_CONT_HI 195
 - HEALTH_CONT_HI_HIS 197
 - HEALTH_CONT_INFO 199
 - HEALTH_DB_HI 201
 - HEALTH_DB_HI_HIS 205
 - HEALTH_DB_HIC 209
 - HEALTH_DB_HIC_HIS 211
 - HEALTH_DB_INFO 214
 - HEALTH_DBM_HI 216
 - HEALTH_DBM_HI_HIS 218
 - HEALTH_DBM_INFO 221
 - HEALTH_GET_ALERT_ACTION_CFG 223
 - HEALTH_GET_ALERT_CFG 226
 - HEALTH_GET_IND_DEFINITION 230
 - HEALTH_TBS_HI 235
 - HEALTH_TBS_HI_HIS 238
 - HEALTH_TBS_INFO 242
 - MQREADALL 249
 - MQREADALLCLOB 252
 - MQRECEIVEALL 259
 - MQRECEIVEALLCLOB 262
 - PD_GET_LOG_MSGS 543
 - SNAP_GET_AGENT 315
 - SNAP_GET_AGENT_MEMORY_POOL 319
 - SNAP_GET_APPL 324
 - SNAP_GET_APPL_INFO 334
 - SNAP_GET_BP 341
 - SNAP_GET_BP_PART 347
 - SNAP_GET_CONTAINER (deprecated) 566
 - SNAP_GET_CONTAINER_V91 351
 - SNAP_GET_DB (deprecated) 568
 - SNAP_GET_DB_MEMORY_POOL 369
 - SNAP_GET_DB_V91 356
 - SNAP_GET_DBM 374
 - SNAP_GET_DBM_MEMORY_POOL 379
 - SNAP_GET_DETAIL_LOG_V91 383
 - SNAP_GET_DYN_SQL (deprecated) 576
 - SNAP_GET_DYN_SQL_V91 387
 - SNAP_GET_FCM 392
 - SNAP_GET_FCM_PART 395
 - SNAP_GET_HADR 398
 - SNAP_GET_LOCK 403
 - SNAP_GET_LOCKWAIT 409
 - SNAP_GET_STMT 415
 - SNAP_GET_STO_PATHS (deprecated) 579
 - SNAP_GET_STORAGE_PATHS 421
 - SNAP_GET_SUBSECTION 425
 - SNAP_GET_SWITCHES 429
 - SNAP_GET_TAB (deprecated) 580
 - SNAP_GET_TAB_REORG 436
 - SNAP_GET_TAB_V91 432
 - SNAP_GET_TBSP (deprecated) 582
 - SNAP_GET_TBSP_PART (deprecated) 586
 - SNAP_GET_TBSP_PART_V91 447
 - SNAP_GET_TBSP_QUIESKER 452
 - SNAP_GET_TBSP_RANGE 456

functions (*continued*)

table functions (*continued*)

SNAP_GET_TBSP_V91 441
SNAP_GET_UTIL 460
SNAP_GET_UTIL_PROGRESS 464
SNAPSHOT_AGENT
(deprecated) 589
SNAPSHOT_APPL
(deprecated) 590
SNAPSHOT_APPL_INFO
(deprecated) 596
SNAPSHOT_BP (deprecated) 599
SNAPSHOT_CONTAINER
(deprecated) 602
SNAPSHOT_DATABASE
(deprecated) 604
SNAPSHOT_DBM
(deprecated) 611
SNAPSHOT_DYN_SQL
(deprecated) 614
SNAPSHOT_FCM
(deprecated) 616
SNAPSHOT_FCMNODE
(deprecated) 618
SNAPSHOT_LOCK
(deprecated) 620
SNAPSHOT_LOCKWAIT
(deprecated) 622
SNAPSHOT_QUIESCERS
(deprecated) 624
SNAPSHOT_RANGES
(deprecated) 626
SNAPSHOT_STATEMENT
(deprecated) 628
SNAPSHOT_SUBJECT
(deprecated) 631
SNAPSHOT_SWITCHES
(deprecated) 633
SNAPSHOT_TABLE
(deprecated) 635
SNAPSHOT_TBREORG
(deprecated) 637
SNAPSHOT_TBS
(deprecated) 639
SNAPSHOT_TBS_CFG
(deprecated) 642
SQLCACHE_SNAPSHOT
(deprecated) 645
supported 8
versus administrative views 3

G

GENERATE_DISTFILE procedure 483
GET STMM TUNING
DBPARTITIONNUM command
using ADMIN_CMD 78
GET_DB_CONFIG table function 563
GET_DBM_CONFIG table function 565
GET_DBSIZE_INFO procedure 539
GET_ROUTINE_OPTS scalar
function 474
GET_ROUTINE_SAR procedure 475
GET_SWRD_SETTINGS procedure 485
groups
retrieving group membership 273

H

health alerts
retrieving alert action
configuration 223
retrieving alert configuration 226
health indicators
retrieving definitions 230
HEALTH_CONT_HI table function 195
HEALTH_CONT_HI_HIS table
function 197
HEALTH_CONT_INFO table
function 199
HEALTH_DB_HI table function 201
HEALTH_DB_HI_HIS table function 205
HEALTH_DB_HIC table function 209
HEALTH_DB_HIC_HIS table
function 211
HEALTH_DB_INFO table function 214
HEALTH_DBM_HI table function 216
HEALTH_DBM_HI_HIS table
function 218
HEALTH_DBM_INFO table
function 221
HEALTH_GET_ALERT_ACTION_CFG
table function 223
HEALTH_GET_ALERT_CFG table
function 226
HEALTH_GET_IND_DEFINITION table
function 230
HEALTH_HI_REC procedure 233
HEALTH_TBS_HI table function 235
HEALTH_TBS_HI_HIS table
function 238
HEALTH_TBS_INFO table function 242
help
displaying 654
for SQL statements 653
history file
retrieve information 527

I

IMPORT command
using ADMIN_CMD 80
Information Center
updating 655
versions 654
viewing in different languages 654
INITIALIZE TAPE command
using ADMIN_CMD 94
installed
retrieving DB2 product
information 191
instance
retrieving current instance
information 189

L

LOAD command
using ADMIN_CMD 96
LOCKS_HELD administrative view 297
LOCKWAIT administrative view 301
LOG_UTILIZATION administrative
view 306

LONG_RUNNING_SQL administrative
view 308

M

MQPUBLISH scalar function 244
MQREAD scalar function 247
MQREADALL table function 249
MQREADALLCLOB table function 252
MQREADCLOB scalar function 255
MQRECEIVE scalar function 257
MQRECEIVEALL table function 259
MQRECEIVEALLCLOB table
function 262
MQRECEIVECLOB scalar function 265
MQSEND scalar function 267
MQSUBSCRIBE scalar function 269
MQUNSUBSCRIBE scalar function 271

N

notices 659
notification lists
retrieving contact list 542
notification log messages
retrieving 543
NOTIFICATIONLIST administrative
view 542

O

OBJECTOWNERS administrative
view 276
objects
retrieving object ownership 276
ordering DB2 books 652

P

PD_GET_LOG_MSGS table function 543
PDLOGMSG_LAST24HOURS
administrative view 543
printed books
ordering 652
privileges
retrieving 278
PRIVILEGES administrative view 278
problem determination
online information 657
retrieving DB2 notification log
messages 543
tutorials 657
procedures 2
ADMIN_CMD 38
ADMIN_COPY_SCHEMA 498
ADMIN_DROP_SCHEMA 503
ADMIN_REMOVE_MSGS 43
ALTOBJ 516
AM_DROP_TASK 22
AM_GET_LOCK_CHN_TB 23
AM_GET_LOCK_CHNS 25
AM_GET_LOCK_RPT 26
AM_GET_RPT 34
AM_SAVE_TASK 36
ANALYZE_LOG_SPACE 480

procedures (*continued*)

- CAPTURE_STORAGEMGMT_INFO 493
- CREATE_STORAGEMGMT_TABLES 495
- deprecated 559
- DROP_STORAGEMGMT_TABLES 497
- GENERATE_DISTFILE 483
- GET_DBSIZE_INFO 539
- GET_ROUTINE_SAR 475
- GET_SWRD_SETTINGS 485
- HEALTH_HI_REC 233
- PUT_ROUTINE_SAR 476
- REBIND_ROUTINE_PACKAGE 478
- REORGCHK_IX_STATS 550
- REORGCHK_TB_STATS 553
- SET_ROUTINE_OPTS 479
- SET_SWRD_SETTINGS 488
- SNAP_WRITE_FILE 313
- SNAPSHOT_FILEW (deprecated) 619
- STEPWISE_REDISTRIBUTE_DBPG 491
- supported 8
- SYSINSTALLOBJECTS 558
- SYSINSTALLROUTINES 647
- PRUNE HISTORY/LOGFILE command using ADMIN_CMD 115
- PUT_ROUTINE_SAR procedure 476

Q

- QUERY_PREP_COST administrative view 311
- QUIESCE DATABASE command using ADMIN_CMD 117
- QUIESCE TABLESPACES FOR TABLE command using ADMIN_CMD 119

R

- REBIND_ROUTINE_PACKAGE procedure 478
- REDISTRIBUTE DATABASE PARTITION GROUP command using ADMIN_CMD 122
- redistributing data procedures 480, 483, 485, 488, 491
- REG_VARIABLES administrative view 187
- registry variables retrieving settings in use 187
- REORG INDEXES/TABLE command using ADMIN_CMD 126
- REORGCHK_IX_STATS procedure 550
- REORGCHK_TB_STATS procedure 553
- RESET ALERT CONFIGURATION command using ADMIN_CMD 136
- RESET DATABASE CONFIGURATION command using ADMIN_CMD 139
- RESET DATABASE MANAGER CONFIGURATION command using ADMIN_CMD 141
- REWIND TAPE command using ADMIN_CMD 143

routines

- SQL administrative supported 8
- SQL administrative routines 559
- RUNSTATS command using ADMIN_CMD 144

S

- scalar functions
 - SQLERRM 555
- schemas
 - copying schemas and objects 498
 - dropping schemas and objects 503
- SET TAPE POSITION command using ADMIN_CMD 156
- SET_ROUTINE_OPTS procedure 479
- SET_SWRD_SETTINGS procedure 488
- SNAP_GET_AGENT table function 315
- SNAP_GET_AGENT_MEMORY_POOL table function 319
- SNAP_GET_APPL table function 324
- SNAP_GET_APPL_INFO table function 334
- SNAP_GET_BP table function 341
- SNAP_GET_BP_PART table function 347
- SNAP_GET_CONTAINER deprecated table function 566
- SNAP_GET_CONTAINER_V91 table function 351
- SNAP_GET_DB deprecated table function 568
- SNAP_GET_DB_MEMORY_POOL table function 369
- SNAP_GET_DB_V91 table function 356
- SNAP_GET_DBM table function 374
- SNAP_GET_DBM_MEMORY_POOL table function 379
- SNAP_GET_DETAIL_LOG_V91 table function 383
- SNAP_GET_DYN_SQL deprecated table function 576
- SNAP_GET_DYN_SQL_V91 table function 387
- SNAP_GET_FCM table function 392
- SNAP_GET_FCM_PART table function 395
- SNAP_GET_HADR table function 398
- SNAP_GET_LOCK table function 403
- SNAP_GET_LOCKWAIT table function 409
- SNAP_GET_STMT table function 415
- SNAP_GET_STO_PATHS deprecated table function 579
- SNAP_GET_STORAGE_PATHS table function 421
- SNAP_GET_SUBSECTION table function 425
- SNAP_GET_SWITCHES table function 429
- SNAP_GET_TAB deprecated table function 580
- SNAP_GET_TAB_REORG table function 436
- SNAP_GET_TAB_V91 table function 432
- SNAP_GET_TBSP deprecated table function 582
- SNAP_GET_TBSP_PART deprecated table function 586
- SNAP_GET_TBSP_PART_V91 table function 447
- SNAP_GET_TBSP_QUIESCER table function 452
- SNAP_GET_TBSP_RANGE table function 456
- SNAP_GET_TBSP_V91 table function 441
- SNAP_GET_UTIL table function 460
- SNAP_GET_UTIL_PROGRESS table function 464
- SNAP_WRITE_FILE procedure 313
- SNAPAGENT administrative view 315
- SNAPAGENT_MEMORY_POOL administrative view 319
- SNAPAPPL administrative view 324
- SNAPAPPL_INFO administrative view 334
- SNAPBP administrative view 341
- SNAPBP_PART administrative view 347
- SNAPCONTAINER administrative view 351
- SNAPDB administrative view 356
- SNAPDB_MEMORY_POOL administrative view 369
- SNAPDBM administrative view 374
- SNAPDBM_MEMORY_POOL administrative view 379
- SNAPDETAILLOG administrative view 383
- SNAPDYN_SQL administrative view 387
- SNAPFCM administrative view 392
- SNAPFCM_PART administrative view 395
- SNAPHADR administrative view 398
- SNAPLOCK administrative view 403
- SNAPLOCKWAIT administrative view 409
- SNAPSHOT_AGENT deprecated table function 589
- SNAPSHOT_APPL deprecated table function 590
- SNAPSHOT_APPL_INFO deprecated table function 596
- SNAPSHOT_BP deprecated table function 599
- SNAPSHOT_CONTAINER deprecated table function 602
- SNAPSHOT_DATABASE deprecated table function 604
- SNAPSHOT_DBM deprecated table function 611
- SNAPSHOT_DYN_SQL deprecated table function 614
- SNAPSHOT_FCM deprecated table function 616
- SNAPSHOT_FCMNODE deprecated table function 618
- SNAPSHOT_FILEW deprecated procedure 619
- SNAPSHOT_LOCK deprecated table function 620
- SNAPSHOT_LOCKWAIT deprecated table function 622

SNAPSHOT QUIESCERS deprecated
 table function 624

SNAPSHOT_RANGES deprecated table
 function 626

SNAPSHOT_STATEMENT deprecated
 table function 628

SNAPSHOT_SUBSECT deprecated table
 function 631

SNAPSHOT_SWITCHES deprecated table
 function 633

SNAPSHOT_TABLE deprecated table
 function 635

SNAPSHOT_TBREORG deprecated table
 function 637

SNAPSHOT_TBS deprecated table
 function 639

SNAPSHOT_TBS_CFG deprecated table
 function 642

SNAPSTMT administrative view 415

SNAPSTORAGE_PATHS administrative
 view 421

SNAPSUBSECTION administrative
 view 425

SNAPSWITCHES administrative
 view 429

SNAPTAB administrative view 432

SNAPTAB_REORG administrative
 view 436

SNAPTbsp administrative view 441

SNAPTbsp_QUIESCER administrative
 view 452

SNAPTbsp_RANGE administrative
 view 456

SNAPTbspPART administrative
 view 447

SNAPUTIL administrative view 460

SNAPUTIL_PROGRESS administrative
 view 464

split mirror
 retrieving database paths 532

SQL administrative routines
 deprecated routines 559

SQL statements
 displaying help 653

SQLCACHE_SNAPSHOT deprecated
 table function 645

SQLERRM scalar function 555

STEPWISE_REDISTRIBUTE_DBPG
 procedure 491

storage management tool
 stored procedures 493, 495, 497

supported functions 8

SYSINSTALLOBJECTS procedure 558

SYSINSTALLROUTINES deprecated
 procedure 647

system information
 retrieving 193

T

table functions
 ADMIN_GET_MSGS 41
 ADMIN_GET_TAB_INFO 506
 AUTH_LIST_GROUPS_FOR_AUTHID
 deprecated 559
 HEALTH_GET_ALERT_ACTION_CFG 223
 HEALTH_GET_ALERT_CFG 226

table functions (*continued*)
 HEALTH_GET_IND_DEFINITION 230
 PD_GET_LOG_MSGS 543
 SNAP_GET_AGENT 315
 SNAP_GET_AGENT_MEMORY_POOL 315
 SNAP_GET_APPL 324
 SNAP_GET_APPL_INFO 334
 SNAP_GET_BP 341
 SNAP_GET_BP_PART 347
 SNAP_GET_CONTAINER_V91 351
 SNAP_GET_DB_MEMORY_POOL 369
 SNAP_GET_DB_V91 356
 SNAP_GET_DBM 374
 SNAP_GET_DBM_MEMORY_POOL 379
 SNAP_GET_DETAIL_LOG_V91 383
 SNAP_GET_DYN_SQL_V91 387
 SNAP_GET_FCM 392
 SNAP_GET_FCM_PART 395
 SNAP_GET_HADR 398
 SNAP_GET_LOCK 403
 SNAP_GET_LOCKWAIT 409
 SNAP_GET_STMT 415
 SNAP_GET_STORAGE_PATHS 421
 SNAP_GET_SUBSECTION 425
 SNAP_GET_SWITCHES 429
 SNAP_GET_TAB_REORG 436
 SNAP_GET_TAB_V91 432
 SNAP_GET_TBSP_PART_V91 447
 SNAP_GET_TBSP_QUIESCER 452
 SNAP_GET_TBSP_RANGE 456
 SNAP_GET_TBSP_V91 441
 SNAP_GET_UTIL 460
 SNAP_GET_UTIL_PROGRESS 464
 supported 8
 versus administrative views 3

tables
 retrieving size and state 506

TBSP_UTILIZATION administrative
 view 467

terms and conditions
 use of publications 658

TOP_DYNAMIC_SQL administrative
 view 472

troubleshooting
 online information 657
 tutorials 657

tutorials
 troubleshooting and problem
 determination 657
 Visual Explain 657

U

UNQUIESCE DATABASE command
 using ADMIN_CMD 157

UPDATE ALERT CONFIGURATION
 command
 using ADMIN_CMD 159

UPDATE CONTACT command
 using ADMIN_CMD 164

UPDATE CONTACTGROUP command
 using ADMIN_CMD 166

UPDATE DATABASE CONFIGURATION
 command
 using ADMIN_CMD 168

UPDATE DATABASE MANAGER
 CONFIGURATION command
 using ADMIN_CMD 171

UPDATE HEALTH NOTIFICATION
 CONTACT LIST command
 using ADMIN_CMD 174

UPDATE HISTORY command
 using ADMIN_CMD 176

UPDATE STMM TUNING
 DBPARTITIONNUM Command
 using ADMIN_CMD 178

updates
 DB2 Information Center 655
 Information Center 655

V

views
 administrative views
 ADMINTABINFO 506
 APPL_PERFORMANCE 286
 APPLICATIONS 280
 AUTHORIZATIONIDS 275
 BP_HITRATIO 288
 BP_READ_IO 290
 BP_WRITE_IO 292
 CONTACTGROUPS 523
 CONTACTS 525
 CONTAINER_UTILIZATION 294
 DB_HISTORY 527
 DBCFG 182
 DBMCFG 184
 DBPATHS 532
 ENV_INST_INFO 189
 ENV_PROD_INFO 191
 ENV_SYS_INFO 193
 LOCKS_HELD 297
 LOCKWAIT 301
 LOG_UTILIZATION 306
 LONG_RUNNING_SQL 308
 NOTIFICATIONLIST 542
 OBJECTOWNERS 276
 PDLOGMSG_LAST24HOURS 543
 PRIVILEGES 278
 QUERY_PREP_COST 311
 REG_VARIABLES 187
 SNAPAGENT 315
 SNAPAGENT_MEMORY_POOL 319
 SNAPAPPL 324
 SNAPAPPL_INFO 334
 SNAPBP 341
 SNAPBP_PART 347
 SNAPCONTAINER 351
 SNAPDB 356
 SNAPDB_MEMORY_POOL 369
 SNAPDBM 374
 SNAPDBM_MEMORY_POOL 379
 SNAPDETAILLOG 383
 SNAPDYN_SQL 387
 SNAPFCM 392
 SNAPFCM_PART 395
 SNAPHADR 398
 SNAPLOCK 403
 SNAPLOCKWAIT 409
 SNAPSTMT 415
 SNAPSTORAGE_PATHS 421
 SNAPSUBSECTION 425

| | |
|---|-----|
| views (<i>continued</i>) | |
| administrative views (<i>continued</i>) | |
| SNAPSWITCHES | 429 |
| SNAPTAB | 432 |
| SNAPTAB_REORG | 436 |
| SNAPTbsp | 441 |
| SNAPTbsp_QUIESCER | 452 |
| SNAPTbsp_RANGE | 456 |
| SNAPTbspPART | 447 |
| SNAPUTIL | 460 |
| SNAPUTIL_PROGRESS | 464 |
| Tbsp_UTILIZATION | 467 |
| TOP_DYNAMIC_SQL | 472 |
| Visual Explain | |
| tutorial | 657 |

Contacting IBM

To contact IBM in your country or region, check the IBM Directory of Worldwide Contacts at <http://www.ibm.com/planetwide>

To learn more about DB2 products, go to <http://www.ibm.com/software/data/db2/>.



Printed in USA

SC10-4293-00



Spine information:

IBM DB2 DB2 Version 9

Administrative SQL Routines and Views

