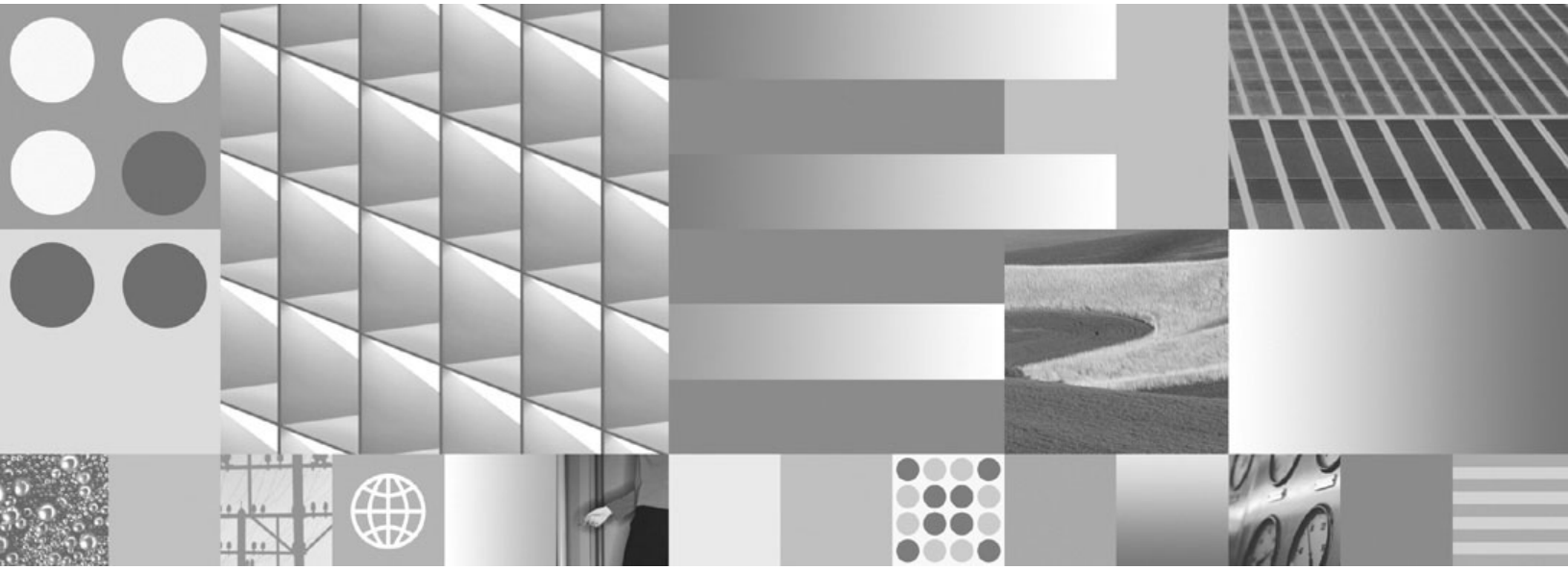
DB2 V9.5
for Linux, UNIX, and Windows

V 9 R 5



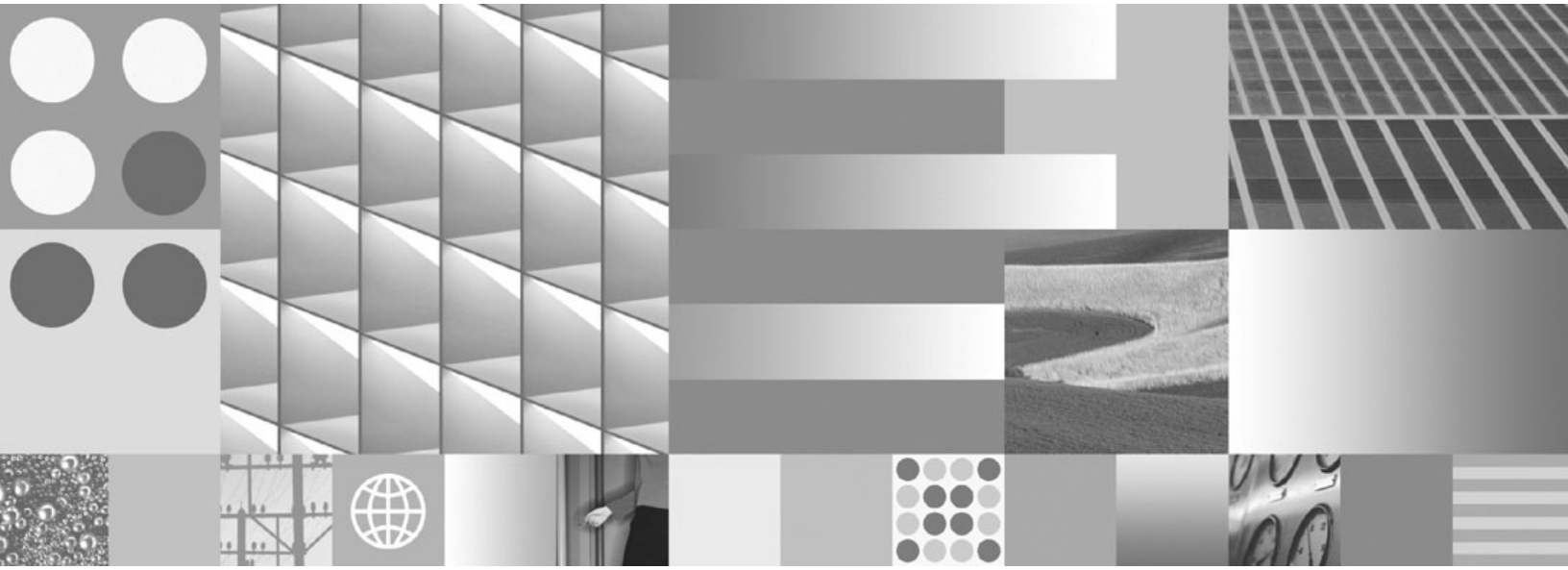
# 数据服务器、数据库和数据库对象指南

2010 年 12 月更新

S151-0612-02

DB2 V9.5
for Linux, UNIX, and Windows

V 9 R 5



# 数据服务器、数据库和数据库对象指南

2010 年 12 月更新

S151-0612-02

**注意**

使用此信息及其支持的产品前，请先阅读第 573 页的附录 B，『声明』下的常规信息。

**修订版声明**

此文档包含 IBM 的所有权信息。它在许可协议中提供，且受版权法的保护。本出版物中包含的信息不包括对任何产品的保证，且提供的任何语句都不需要如此解释。

您可在线或通过当地的 IBM 代表处订购 IBM 出版物。

- 要在线订购出版物，请转至 IBM 出版物中心，网址为: www.ibm.com/shop/publications/order
- 要查找当地的 IBM 代表处，请转至 IBM 全球联系人目录，网址为: www.ibm.com/planetwide

要从美国或加拿大的 DB2 市场和销售部订购 DB2 出版物，请致电 1-800-IBM-4YOU（426-4968）。

当您向 IBM 发送信息时，即同意授予 IBM 独一无二的权力以它认为适当且不会对您造成任何影响的方式使用或分发该信息。

**© Copyright IBM Corporation 1993, 2010.**

# 目录


关于本书 . . . . . ix

第 1 部分 数据服务器 . . . . . 1

第 1 章 DB2 数据服务器 . . . . . 3
数据服务器容量的管理 . . . . . 3
启用大内存页支持（AIX） . . . . . 4
固定 DB2 数据库共享内存（AIX） . . . . . 5

第 2 章 多个 DB2 副本 . . . . . 7
缺省 IBM 数据库客户机接口副本 . . . . . 7
运行多个 DB2 副本时设置 DAS . . . . . 10
使用多个 DB2 副本时设置缺省实例（Windows） . . 11
数据库管理器的多个实例 . . . . . 12
多个实例（Windows） . . . . . 12
更新 DB2 副本（Windows） . . . . . 13
同时运行多个实例（Windows） . . . . . 14
使用同一 DB2 副本或不同 DB2 副本上的实例. . . 14

第 3 章 自主计算 . . . . . 17
自动功能 . . . . . 17
自动维护 . . . . . 18
维护时间段 . . . . . 19
自调整内存 . . . . . 19
DB2 中的内存分配 . . . . . 20
自调整内存操作的详细信息和局限性 . . . . . 23
内存参数交互和局限性 . . . . . 23
启用自调整内存功能 . . . . . 25
禁用自调整内存功能 . . . . . 26
确定启用了自调整功能的内存使用者 . . . . . 26
分区数据库环境中的自调整内存 . . . . . 27
在分区数据库环境中使用自调整内存 . . . . . 28
配置内存和内存堆 . . . . . 30
代理程序和进程技术模型配置 . . . . . 32
代理程序、进程模型和内存配置 . . . . . 32
自动存储器 . . . . . 34
自动存储器表空间 . . . . . 34
自动存储器数据库 . . . . . 39
自动存储器限制 . . . . . 42
创建压缩字典 . . . . . 42
行压缩 . . . . . 44
自动收集统计信息 . . . . . 45
启用自动收集统计信息 . . . . . 49
配置顾问程序 . . . . . 49
使用“配置顾问程序”调整配置参数 . . . . . 50
生成数据库配置建议 . . . . . 50
示例: 使用“配置顾问程序”请求配置建议 . . . . 51
实用程序调速 . . . . . 53
异步索引清除 . . . . . 53
MDC 表的异步索引清除 . . . . . 54

第 4 章 实例 . . . . . 57
设计实例 . . . . . 58
缺省实例 . . . . . 59
实例目录 . . . . . 59
多个实例（Linux 和 UNIX） . . . . . 60
多个实例（Windows） . . . . . 61
创建实例 . . . . . 61
修改实例 . . . . . 62
更新实例配置（Linux 和 UNIX） . . . . . 63
更新实例配置（Windows） . . . . . 64
使用实例 . . . . . 64
自动启动实例 . . . . . 64
启动实例（Linux 和 UNIX） . . . . . 65
启动实例（Windows） . . . . . 65
连接至实例和从实例拆离 . . . . . 66
使用同一 DB2 副本或不同 DB2 副本上的实例. . 66
停止实例（Linux 和 UNIX） . . . . . 66
停止实例（Windows） . . . . . 67
删除实例 . . . . . 68

第 5 章 轻量级目录访问协议（LDAP） 71
LDAP 环境中的安全性注意事项 . . . . . 71
DB2 使用的 LDAP 对象类和属性 . . . . . 72
使用 DB2 对象类和属性来扩展 LDAP 目录模式 . . 81
受支持的 LDAP 客户机和服务器配置 . . . . . 82
LDAP 支持和 DB2 Connect . . . . . 82
扩展 IBM Tivoli Directory Server 的目录模式 . . 83
Netscape LDAP 目录支持和属性定义 . . . . . 84
展开 Sun One Directory Server 的目录模式 . . . 86
Windows Active Directory . . . . . 88
完成 DB2 安装之后启用 LDAP 支持 . . . . . 90
注册 LDAP 条目 . . . . . 91
安装之后注册 DB2 服务器 . . . . . 91
编目节点别名以进行连接 (ATTACH) . . . . . 92
在 LDAP 目录中注册数据库 . . . . . 92
注销 LDAP 条目 . . . . . 93
注销 DB2 服务器 . . . . . 93
从 LDAP 目录中注销数据库 . . . . . 93
配置 LDAP 用户 . . . . . 93
创建 LDAP 用户 . . . . . 93
为 DB2 应用程序配置 LDAP 用户. . . . . 94
在 LDAP 环境中设置用户级的 DB2 注册表变量 94
禁用 LDAP 支持 . . . . . 95
更新 DB2 服务器的协议信息. . . . . 95
将 LDAP 客户机重新路由至另一台服务器 . . . . 95
在 LDAP 环境中连接至远程服务器 . . . . . 96
刷新本地数据库和节点目录中的 LDAP 条目 . . . 96
搜索 LDAP 服务器 . . . . . 97

第 2 部分 数据库 . . . . . 99


© Copyright IBM Corp. 1993, 2010

**第 6 章 数据库 . . . . . 101**
设计数据库 . . . . . 101
数据库目录和文件 . . . . . 102
数据库对象的空间需求 . . . . . 108
日志文件的空间需求 . . . . . 109
轻量级目录访问协议（LDAP）目录服务 . . . . 110
创建数据库 . . . . . 110
自动存储器数据库 . . . . . 112
编目数据库 . . . . . 117
将实用程序绑定至数据库 . . . . . 118
创建数据库别名 . . . . . 118
连接至分布式关系数据库 . . . . . 119
分布式关系数据库的远程工作单元 . . . . . 120
应用程序导向的分布式工作单元 . . . . . 123
应用程序进程连接状态 . . . . . 124
连接状态 . . . . . 125
控制工作单元语义的选项 . . . . . 126
数据表示注意事项 . . . . . 126
查看本地或系统数据库目录文件 . . . . . 127
删除数据库 . . . . . 127
删除别名 . . . . . 127

**第 7 章 数据库分区 . . . . . 129**

**第 8 章 缓冲池 . . . . . 131**
设计缓冲池 . . . . . 131
缓冲池内存保护（在 POWER6 上运行的 AIX） . . 133
创建缓冲池 . . . . . 134
修改缓冲池 . . . . . 135
删除缓冲池 . . . . . 136

**第 9 章 表空间 . . . . . 137**
设计表空间 . . . . . 138
表空间的类型 . . . . . 139
SMS 和 DMS 表空间的比较 . . . . . 152
为表选择表空间时的注意事项 . . . . . 154
自动重新调整表空间的大小 . . . . . 155
添加或删除容器时调整预取大小 . . . . . 158
不使用文件系统高速缓存的表空间 . . . . . 160
表空间扩展数据块大小 . . . . . 165
表空间的页大小 . . . . . 167
表空间磁盘 I/O . . . . . 167
定义初始表空间 . . . . . 168
连接 DMS 直接磁盘访问设备 . . . . . 169
配置和设置 DMS 直接磁盘访问（Linux） . . . 170
创建表空间 . . . . . 172
改变表空间 . . . . . 175
改变 SMS 表空间 . . . . . 176
改变 DMS 表空间 . . . . . 176
改变自动存储器表空间 . . . . . 187
重命名表空间 . . . . . 187
将表空间从脱机状态切换至联机状态 . . . . . 188
当数据位于 RAID 设备上时优化表空间性能 . . . 188
删除表空间 . . . . . 189

**第 10 章 模式 . . . . . 191**
设计模式 . . . . . 192
根据模式将对象分组 . . . . . 194
模式名限制和建议 . . . . . 195
创建模式 . . . . . 195
复制模式 . . . . . 196
使用 ADMIN_COPY_SCHEMA 过程的模式复制的示例 . . . . . 197
使用 db2move 实用程序的模式复制的示例 . . . 198
重新启动失败的复制模式操作 . . . . . 199
删除模式 . . . . . 201

**第 3 部分 数据库对象 . . . . . 203**

**第 11 章 表 . . . . . 205**
表的类型 . . . . . 205
设计表 . . . . . 207
表设计概念 . . . . . 207
表的空间需求 . . . . . 214
用户表数据的空间需求 . . . . . 216
表压缩 . . . . . 217
乐观锁定 . . . . . 223
表分区和数据组织方案 . . . . . 230
创建表 . . . . . 230
声明全局临时表 . . . . . 231
创建类似于现有表的表 . . . . . 231
为登台数据创建表 . . . . . 232
修改表 . . . . . 233
改变具体化查询表属性 . . . . . 233
刷新具体化查询表中的数据 . . . . . 233
更改列属性 . . . . . 234
重命名表 . . . . . 236
恢复不可用摘要表 . . . . . 236
查看表定义 . . . . . 236
表或视图别名 . . . . . 237
删除表 . . . . . 237
删除具体化查询表或登台表 . . . . . 238
表方案和示例 . . . . . 238
方案：乐观锁定和基于时间的检测 . . . . . 238

**第 12 章 约束 . . . . . 243**
约束的类型 . . . . . 243
NOT NULL 约束 . . . . . 244
唯一约束 . . . . . 244
主键约束 . . . . . 245
（表）检查约束 . . . . . 245
外键（引用）约束 . . . . . 245
参考约束 . . . . . 249
设计约束 . . . . . 249
设计唯一约束 . . . . . 249
设计主键约束 . . . . . 250
设计检查约束 . . . . . 250
设计外键（引用）约束 . . . . . 251
设计参考约束 . . . . . 256
创建和修改约束 . . . . . 258
查看表的约束定义 . . . . . 259
删除约束 . . . . . 259

**第 13 章 索引 . . . . . 263**
索引的类型 . . . . . 265
设计索引 . . . . . 266
用于设计索引的工具 . . . . . 267
索引的空间需求 . . . . . 268
创建索引 . . . . . 271
修改索引 . . . . . 272
重命名索引 . . . . . 272
重建索引 . . . . . 272
删除索引 . . . . . 273

**第 14 章 触发器 . . . . . 275**
触发器的类型 . . . . . 276
前触发器 . . . . . 276
后触发器 . . . . . 277
INSTEAD OF 触发器 . . . . . 277
设计触发器 . . . . . 278
指定使触发器触发的对象（触发语句或事件） . . 280
指定触发器触发的时间（BEFORE、AFTER 和 INSTEAD OF 子句） . . . . . 281
定义触发器操作将触发的条件（WHEN 子句） 283
触发器中受支持的 SQL PL 语句 . . . . . 284
使用转换变量访问触发器中的旧列值和新列值 . . 285
使用转换表引用旧表结果集和新表结果集 . . . 285
创建触发器 . . . . . 287
修改和删除触发器 . . . . . 288
触发器和触发器用法的示例 . . . . . 288
触发器与引用约束之间的交互的示例 . . . . . 288
使用触发器定义操作的示例 . . . . . 290
使用触发器定义业务规则的示例 . . . . . 291
使用触发器防止对表进行操作的示例 . . . . . 292

**第 15 章 序列 . . . . . 293**
设计序列 . . . . . 293
管理序列行为 . . . . . 294
应用程序性能和序列 . . . . . 295
比较序列与标识列 . . . . . 295
创建序列 . . . . . 296
生成顺序值 . . . . . 296
确定何时使用标识列或序列 . . . . . 297
修改序列 . . . . . 298
查看序列定义 . . . . . 298
删除序列 . . . . . 299
如何编码序列的示例 . . . . . 299
序列引用 . . . . . 300

**第 16 章 视图 . . . . . 305**
设计视图 . . . . . 306
系统目录视图 . . . . . 306
使用检查选项的视图 . . . . . 306
可删除视图 . . . . . 309
可插入视图 . . . . . 309
可更新视图 . . . . . 310
只读视图 . . . . . 310
创建视图 . . . . . 310
创建使用用户定义的函数（UDF）的视图 . . . 311
修改带类型视图 . . . . . 312
恢复不可用视图 . . . . . 312
删除视图 . . . . . 313

**第 4 部分 参考 . . . . . 315**

**第 17 章 符合命名规则 . . . . . 317**
命名规则 . . . . . 317
DB2 对象命名规则 . . . . . 318
定界标识和对象名 . . . . . 319
用户、用户标识和组命名规则 . . . . . 320
NLS 环境中的命名规则 . . . . . 320
Unicode 环境中的命名规则 . . . . . 321

**第 18 章 SQL 和 XML 限制 . . . . . 323**

**第 19 章 注册表变量和环境变量 . . . . . 333**
环境变量和概要文件注册表 . . . . . 333
声明、显示、更改、重新设置和删除注册表变量和环境变量 . . . . . 335
在 Windows 上设置环境变量 . . . . . 337
在 Linux 和 UNIX 操作系统上设置环境变量 . . 338
设置当前实例环境变量 . . . . . 339
聚集注册表变量 . . . . . 340
DB2 注册表变量和环境变量 . . . . . 341
常规注册表变量 . . . . . 346
系统环境变量 . . . . . 354
通信变量 . . . . . 362
命令行变量 . . . . . 364
分区数据库环境变量 . . . . . 366
查询编译器变量 . . . . . 368
性能变量 . . . . . 372
其他变量 . . . . . 389

**第 20 章 配置参数 . . . . . 407**
使用配置参数配置 DB2 数据库管理器 . . . . . 408
配置参数摘要 . . . . . 411
影响代理程序数的配置参数 . . . . . 422
影响查询优化的配置参数 . . . . . 423
配置 max_coordagents 和 max_connections 时的限制和行为 . . . . . 425
数据库管理器配置参数 . . . . . 426
agent_stack_sz - 代理程序堆栈大小 . . . . . 426
agentpri - 代理程序的优先级 . . . . . 428
aslheapsz - 应用程序支持层堆大小 . . . . . 429
audit_buf_sz - 审计缓冲区大小 . . . . . 430
authentication - 认证类型 . . . . . 431
catalog_noauth - 没有权限时允许的编目 . . . . 432
clnt_krb_plugin - 客户机 Kerberos 插件 . . . . 433
clnt_pw_plugin - 客户机用户标识密码插件 . . . 433
cluster_mgr - 集群管理器名称 . . . . . 433
comm_bandwidth - 通信带宽 . . . . . 434
conn_elapse - 连接耗用时间 . . . . . 435
cpuspeed - CPU 速度 . . . . . 435
dft_account_str - 缺省对方付费帐户 . . . . . 436
dft_monswitches - 缺省数据库系统监视器开关 436

dftdbpath - 缺省数据库路径 . . . . . 437
diaglevel -"诊断错误捕获级别" . . . . . 438
diagpath -"诊断数据目录路径" . . . . . 439
dir_cache - 目录高速缓存支持 . . . . . 442
discover - 发现方式 . . . . . 443
discover_inst - 发现服务器实例. . . . . 443
fcm_num_buffers - FCM 缓冲区数 . . . . . 444
fcm_num_channels - FCM 通道数 . . . . . 445
fed_noauth - 绕过联合认证 . . . . . 446
federated - 联合数据库系统支持 . . . . . 446
federated_async - 每个查询的最大异步 TQ 数配置参数. . . . . 446
fenced_pool - 最大受防护进程数 . . . . . 447
group_plugin - 组插件. . . . . 448
health_mon - 运行状况监视器 . . . . . 449
indexrec - 索引重新创建时间 . . . . . 449
instance_memory - 实例内存 . . . . . 451
intra_parallel - 启用分区内并行性 . . . . . 453
java_heap_sz - 最大 Java 解释器堆大小. . . . 453
jdk_path - Java 软件开发者工具箱安装路径 . . 454
keepfenced - 保留受防护进程 . . . . . 455
local_gssplugin - 用于实例级本地授权的 GSS API 插件 . . . . . 455
max_connections - 最大客户机连接数. . . . . 456
max_connretries - 节点连接重试次数 . . . . . 456
max_coordagents - 最大协调代理程序数 . . . . 457
max_querydegree - 最大查询并行度 . . . . . 458
max_time_diff - 节点间的最大时差 . . . . . 458
maxagents - 最大代理程序数 . . . . . 459
maxcagents - 最大并发代理程序数. . . . . 460
mon_heap_sz -"数据库系统监视器堆大小" . . . 460
nodetype - 机器节点类型. . . . . 461
notifylevel - 通知级别. . . . . 462
num_initagents - 池中的初始代理程序数 . . . . 463
num_initfenced - 受防护进程的初始数目. . . . 463
num_poolagents - 代理程序池大小. . . . . 464
numdb - 同时处于活动状态的数据库（包括主机和 System i 数据库）的最大数目 . . . . . 465
query_heap_sz - 查询堆大小. . . . . 465
release - 配置文件发行版级别 . . . . . 466
resync_interval - 事务再同步时间间隔 . . . . 467
rqrioblk - 客户机 I/O 块大小 . . . . . 467
sheapthres -"排序堆阈值". . . . . 468
spm_log_file_sz - 同步点管理器日志文件大小 . . 469
spm_log_path - 同步点管理器日志文件路径. . . 470
spm_max_resync - 同步点管理器再同步代理程序限制 . . . . . 470
spm_name - 同步点管理器名 . . . . . 471
srvcon_auth - 用于服务器中的入局连接的认证类型 . . . . . 471
srvcon_gssplugin_list - 用于服务器中的入局连接的 GSS API 插件列表 . . . . . 471
srvcon_pw_plugin - 用于服务器中的入局连接的用户标识密码插件. . . . . 472
srv_plugin_mode - 服务器插件方式 . . . . . 473
start_stop_time - 启动和停止超时 . . . . . 473
svcename - TCP/IP 服务名称 . . . . . 474
sysadm_group - 系统管理权限组名 . . . . . 474
sysctrl_group - 系统控制权限组名. . . . . 475
sysmaint_group - 系统维护权限组名 . . . . . 475
sysmon_group - 系统监视权限组名 . . . . . 476
tm_database - 事务管理器数据库名称. . . . . 477
tp_mon_name - 事务处理器监视器名 . . . . . 477
trust_allclnts - 信赖所有客户机 . . . . . 479
trust_clntauth - 可信客户机认证 . . . . . 479
util_impact_lim - 实例影响策略. . . . . 480
数据库配置参数. . . . . 481
alt_collate - 备用整理顺序 . . . . . 481
app_ctl_heap_sz - 应用程序控制堆大小 . . . . 481
appgroup_mem_sz - 应用程序组内存集的最大大小 482
appl_memory - 应用程序内存配置参数 . . . . 483
applheapsz - 应用程序堆大小 . . . . . 484
archretrydelay - 出错时的归档重试延迟 . . . . 485
auto_del_rec_obj - 自动删除恢复对象配置参数 485
auto_maint - 自动维护 . . . . . 485
autorestart - 允许自动重新启动. . . . . 487
avg_appls - 平均活动应用程序数 . . . . . 488
backup_pending - 备份暂挂指示器. . . . . 488
blk_log_dsk_ful - 磁盘已满时阻止进行日志记录 489
blocknonlogged -"禁止创建允许不进行日志记录的活动的表" . . . . . 489
catalogcache_sz - 目录高速缓存大小 . . . . . 490
chngpgs_thresh - 更改的页数阈值 . . . . . 491
codepage - 用于数据库的代码页 . . . . . 491
codeset - 用于数据库的代码集 . . . . . 491
collate_info - 整理信息 . . . . . 492
country/region - 数据库地域代码 . . . . . 492
database_consistent - 数据库处于一致状态 . . . 493
database_level - 数据库发行版级别 . . . . . 493
database_memory - 数据库共享内存大小. . . . 493
db_mem_thresh - 数据库内存阈值 . . . . . 495
dbheap - 数据库堆. . . . . 496
decflt_rounding - 十进制浮点数舍入配置参数 . . 497
dft_degree - 缺省度 . . . . . 498
dft_extent_sz - 表空间的缺省扩展数据块大小 . . 499
dft_loadrec_ses - 缺省装入恢复会话数 . . . . 499
dft_mttb_types - 为优化缺省维护的表类型 . . . 500
dft_prefetch_sz - 缺省预取大小. . . . . 500
dft_queryopt - 缺省查询优化类 . . . . . 501
dft_refresh_age - 缺省刷新持续时间 . . . . . 502
dft_sqlmathwarn - 出现算术异常时继续 . . . . 502
discover_db - 发现数据库 . . . . . 503
dlchktime - 检查死锁的时间间隔 . . . . . 504
dyn_query_mgmt - 动态 SQL 和 XQuery 查询管理 . . . . . 504
enable_xmlchar - 对 XML 启用配置参数转换 . . 505
failarchpath - 故障转移日志归档路径 . . . . . 505
groupheap_ratio - 用于应用程序组堆的内存百分比 506
hadr_db_role - HADR 数据库角色. . . . . 506
hadr_local_host - HADR 本地主机名 . . . . . 506
hadr_local_svc - HADR 本地服务名称 . . . . 507
hadr_peer_window - HADR 对等窗口配置参数 507

hadr_remote_host - HADR 远程主机名 . . . . 508
hadr_remote_inst - 远程服务器的 HADR 实例名 508
hadr_remote_svc - HADR 远程服务名称. . . . 509
hadr_syncmode - 处于对等状态的日志写操作的 HADR 同步方式 . . . . . 509
hadr_timeout - HADR 超时值 . . . . . 510
indexrec - 索引重新创建时间 . . . . . 510
jdk_64_path - 64 位 Java 软件开发者工具箱安装路径 DAS . . . . . 512
locklist -“锁定列表的最大存储量” . . . . . 512
locktimeout - 锁定超时 . . . . . 515
log_retain_status - 日志保留状态指示器 . . . . 516
logarchmeth1 - 主日志归档方法 . . . . . 516
logarchmeth2 - 辅助日志归档方法 . . . . . 517
logarchopt1 - 主日志归档选项 . . . . . 518
logarchopt2 - 辅助日志归档选项 . . . . . 518
logbufsz - 日志缓冲区大小 . . . . . 519
logfilsiz - 日志文件大小 . . . . . 520
loghead - 第一个活动日志文件. . . . . 521
logindexbuild - 创建的日志索引页数 . . . . . 521
logpath - 日志文件的位置 . . . . . 521
logprimary - 主日志文件数 . . . . . 522
logretain -“启用日志保留” . . . . . 523
logsecond - 辅助日志文件数. . . . . 523
max_log - 每个事务的最大日志 . . . . . 524
maxappls - 最大活动应用程序数 . . . . . 525
maxfilop - 每个应用程序打开的最大数据库文件数 526
maxlocks -“锁定升级前锁定列表的最大百分比” 526
min_dec_div_3 - 十进制除法，小数位为 3 的 528
mincommit - 要分组的落实数 . . . . . 529
mirrorlogpath - 镜像日志路径 . . . . . 530
multipage_alloc - 启用多页文件分配 . . . . . 531
newlogpath - 更改数据库日志路径. . . . . 531
num_db_backups - 数据库备份数 . . . . . 532
num_freqvalues - 保留的高频值数目 . . . . . 533
num_iocleaners - 异步页清除程序的数目. . . . 534
num_ioservers -“I/O 服务器数” . . . . . 535
num_log_span - 跨越的日志数 . . . . . 536
num_quantiles - 列的分位数. . . . . 536
numarchretry - 出错时重试次数. . . . . 537
numsegs - 缺省 SMS 容器数 . . . . . 537
number_compat -“数字兼容性”数据库配置参数 538
overflowlogpath - 溢出日志路径 . . . . . 538
pagesize - 数据库缺省页大小 . . . . . 539
pckcachesz - 程序包高速缓存大小. . . . . 539
priv_mem_thresh - 专用内存阈值 . . . . . 541
rec_his_retentn - 恢复历史记录保留期 . . . . 541
restore_pending - 复原暂挂 . . . . . 542
restrict_access - 数据库具有受限访问配置参数 542
rollfwd_pending - 前滚暂挂指示器. . . . . 542
self_tuning_mem - 自调整内存 . . . . . 543
seqdetect - 顺序检测标志. . . . . 544
sheapthres_shr - 共享排序的排序堆阈值 . . . . 545
softmax - 恢复范围和软检查点时间间隔. . . . 546
sortheap - 排序堆大小. . . . . 547
stat_heap_sz -“统计信息堆大小” . . . . . 548
stmtheap -“语句堆大小” . . . . . 549
territory - 数据库地域. . . . . 550
trackmod - 启用跟踪已修改页 . . . . . 550
tsm_mgmtclass - Tivoli Storage Manager 管理类 550
tsm_nodename - Tivoli Storage Manager 节点名 551
tsm_owner - Tivoli Storage Manager 所有者名称 551
tsm_password - Tivoli Storage Manager 密码 . . 552
user_exit_status - 用户出口状态指示器 . . . . 552
userexit -“启用用户出口”. . . . . 552
util_heap_sz - 实用程序堆大小 . . . . . 553
varchar2_compat -“Varchar2 兼容性”数据库配置参数 . . . . . 553
vendoropt - 供应商选项 . . . . . 553
wlm_collect_int - 工作负载管理收集时间间隔配置参数 . . . . . 554
DB2 管理服务器（DAS）配置参数 . . . . . 555
authentication - 认证类型 DAS. . . . . 555
contact_host - 联系人列表的位置 . . . . . 555
das_codepage - DAS 代码页 . . . . . 556
das_territory - DAS 地域. . . . . 556
dasadm_group - DAS 管理权限组名 . . . . . 556
db2system - DB2 服务器系统的名称 . . . . . 557
diaglevel -“诊断错误捕获级别”配置参数 . . . . 557
discover - DAS 发现方式 . . . . . 558
exec_exp_task - 执行已到期任务 . . . . . 558
jdk_path - Java 软件开发者工具箱安装路径 DAS 559
sched_enable - 调度程序方式 . . . . . 559
sched_userid - 调度程序用户标识 . . . . . 560
smtp_server - SMTP 服务器. . . . . 560
toolscat_db - 工具目录数据库 . . . . . 560
toolscat_inst - 工具目录数据库实例 . . . . . 561
toolscat_schema - 工具目录数据库模式 . . . . 561

**第 5 部分 附录 . . . . . 563**

**附录 A. DB2 技术信息概述 . . . . . 565**
硬拷贝或 PDF 格式的 DB2 技术库 . . . . . 565
订购印刷版的 DB2 书籍. . . . . 568
从命令行处理器显示 SQL 状态帮助 . . . . . 569
访问不同版本的 DB2 信息中心 . . . . . 569
在 DB2 信息中心中以您的首选语言显示主题: . . 569
更新安装在您的计算机或内部网服务器上的 DB2 信息中心. . . . . 570
DB2 教程. . . . . 571
DB2 故障诊断信息. . . . . 572
条款和条件 . . . . . 572

**附录 B. 声明 . . . . . 573**

**索引 . . . . . 577**

# 关于本书

《数据服务器、数据库和数据库对象指南》提供了使用和管理 DB2® 关系数据库管理系统（RDBMS）产品所需的信息。本书包含有关数据库规划和设计以及数据库对象的实现和管理的信息。本书还包含有关数据库配置和调整的参考信息。

## 本书适用对象

本书主要适用于需要设计、实现和维护数据库以供本地或远程客户端访问的数据库管理员和系统管理员。需要了解 DB2 关系数据库管理系统的管理和操作的程序员和其他用户也可使用本书。

## 本书的结构

本书分为如下四个部分:

**第 1 部分 数据服务器**

本节简要描述 DB2 数据服务器，同时描述了如何管理它们在 AIX® 上 64 位环境中的容量和大页面支持。此外，本节还提供了有关在单台计算机上运行多个 DB2 副本的信息、有关帮助您管理数据库系统的自动功能部件的信息、有关设计、创建和使用实例的信息以及有关配置轻量级目录访问协议（LDAP）服务器的可选信息。

**第 2 部分 数据库**

本节描述数据库、缓冲池、表空间和模式的设计、创建和维护。有关数据库分区的详细信息可在新*《分区和集群指南》*中找到。

**第 3 部分 数据库对象**

本节描述以下数据库对象的设计、创建和维护: 表、约束、索引、触发器、序列和视图。

**第 4 部分 参考**

本节包含有关使用环境变量、注册表变量和配置参数来配置和调整数据库系统的参考信息。本节还列示了各种命名规则以及 SQL 限制和 XML 限制。

© Copyright IBM Corp. 1993, 2010

# 第 1 部分 数据服务器

© Copyright IBM Corp. 1993, 2010

# 第 1 章 DB2 数据服务器

数据服务器提供软件服务以便安全、高效地管理结构化信息。DB2 是混合关系数据和 XML 数据服务器。

数据服务器是指安装了 DB2 数据库引擎的机器。DB2 引擎是健壮而且功能全面的数据库管理系统，它包含根据实际数据库使用情况进行优化的 SQL 支持以及用于帮助管理数据的工具。

IBM 提供了许多数据服务器产品，包括可以访问所有数据服务器的数据服务器客户机。有关 DB2 数据服务器产品、可用的功能部件以及详细的描述和规范的完整列表，请参阅 http://www-306.ibm.com/software/data/db2/9/。

## 数据服务器容量的管理

如果数据服务器的容量不能满足目前或将来的要求，那么可通过增大磁盘空间和创建其他容器或通过增加内存来扩充其容量。如果这些简单措施不能增加所需的容量，还可以考虑添加处理器或物理分区。当通过更改环境来调整系统时，应意识到这类更改可给数据库过程（如装入数据、备份和复原数据库）带来的影响。

### 添加处理器

如果具有单个处理器的单一分区数据库配置已被最大限度地使用，那么可能需要添加处理器或数据库分区。添加处理器可以获得更大的处理能力。在 SMP 系统中，多个处理器共享内存和存储系统资源。由于所有处理器均在一个系统内，因此不存在附加的开销费用（如铺设系统间的通信线路），也不会增加系统间的协调任务。一些实用程序（例如，装入、备份和复原）可以利用其他处理器。

**注:** 某些操作系统（例如，Solaris 操作系统）可以动态地使处理器联机和脱机。

如果添加了处理器，请检查并修改那些决定处理器使用数目的数据库配置参数。下列数据库配置参数决定处理器的使用数目，可能需要进行更新:

- 缺省级别 (dft_degree)
- 最大并行度 (max_querydegree)
- 启用分区内并行性 (intra_parallel)

此外，还应评估那些决定应用程序如何执行并行处理的参数。

在使用 TCP/IP 进行通信的环境中，应检查 DB2TCPCONNMGRS 注册表变量的值。

### 添加物理分区

如果数据库管理器当前在分区数据库环境中，那么可以通过添加不同的单处理器物理分区或多处理器物理分区来增加数据存储空间和处理能力。每个数据库分区上的内存和存储系统资源不与其他数据库分区共享。虽然添加数据库分区可能会导致通信和任务协调问题，但这一选择所带来的好处是可在多个系统上平衡数据和用户访问。数据库管理器支持此环境。

© Copyright IBM Corp. 1993, 2010

无论数据库管理器系统处于运行状态还是停止状态，均可添加数据库分区。但是，如果在系统运行时添加数据库分区，那么必须在停止并重新启动系统后，才能将数据库迁移到新的数据库分区。

当添加一个新的数据库分区时，在该过程完成并将新服务器成功集成到系统中之前，不能删除或创建使用新数据库分区的数据库。

## 启用大内存页支持（AIX）

在 AIX 操作系统上，要在 DB2 数据库系统中启用大内存页支持，必须配置某些操作系统参数，然后设置 **DB2_LARGE_PAGE_MEM** 注册表变量。

### 开始之前

您必须具有 root 用户权限才能使用 AIX 操作系统命令。

### 关于本任务

除了传统的 4 KB 页大小以外，IBM® eServer™ pSeries® 系统中的 POWER4™ 和更高级别的处理器还支持 16 MB 页大小。需要密集访问内存并且使用大量虚拟内存的应用程序（例如，IBM DB2 AIX 版版本 9.5）可以通过使用大内存页来提高性能。

**注:**

1. 设置 **DB2_LARGE_PAGE_MEM** 注册表变量还指示已将内存固定。
2. 在当配置系统以固定内存和支持大内存页时，请您务必极为小心谨慎。固定太多的内存将导致未固定的内存页执行繁重的页面调度活动。如果没有足够的内存来支持 4 KB 页，分配太多物理内存给大内存页将使系统性能降级。

### 限制

启用大内存页可以防止自调整内存管理器自动调整数据库内存消耗总量，因此，只应对具有相对静态数据库内存需求的标准定义的工作负载启用大内存页。

### 过程

在 AIX 操作系统上，要在 DB2 数据库系统中启用大内存页支持:

1. 通过发出 vmo 命令并指定下列标志来为 AIX 服务器配置大内存页支持:

   ```
   vmo -r -o lgpg_size=LargePageSize -o lgpg_regions=LargePages
   ```

   其中 *LargePageSize* 指定硬件支持的大内存页的大小（以字节计），而 *LargePages* 指定要保留的大内存页的数目。 例如，如果需要为大内存页支持分配 25 GB，那么按如下方式运行命令:

   ```
   vmo -r -o lgpg_size=16777216 -o lgpg_regions=1600
   ```

   有关如何运行 vmo 命令的详细指示信息，请参阅 AIX 手册。
2. 运行 bosboot 命令，以使先前运行的 vmo 命令将在系统下一次引导之后生效。
3. 在服务器启动之后，使它能够固定内存。 发出 vmo 命令并指定下列标志:

   ```
   vmo -o v_pinshm=1
   ```
4. 使用 db2set 命令将 **DB2_LARGE_PAGE_MEM** 注册表变量设置为 DB，然后启动 DB2 数据库管理器。 例如:

```
db2set DB2_LARGE_PAGE_MEM=DB
   db2start
```

### 结果

完成这些步骤之后，DB2 数据库系统将指示操作系统对数据库共享内存区域使用大内存页内存。

## 固定 DB2 数据库共享内存（AIX）

要固定 AIX 操作系统上的 DB2 数据库共享内存，必须配置某些操作系统参数，然后设置 **DB2_PINNED_BP** 注册表变量。

### 开始之前

您必须具有 root 用户权限才能执行 AIX 操作系统命令。

### 关于本任务

固定某些部分的内存的优点在于 - 当您访问被固定的页时，无需执行页替换算法就可以检索该页。其缺点在于 - 您必须小心确保未过度使用系统，否则操作系统的内存管理灵活性会降低。固定太多的内存将导致未固定的内存页执行繁重的页面调度活动。

### 限制

如果您将 **DB2_PINNED_BP** 注册表变量设置为 YES，那么不能启用对数据库共享内存的自调整。

### 过程

要固定 AIX 操作系统上的 DB2 数据库共享内存:

1. 配置 AIX 操作系统以启用固定内存。 发出 vmo 命令并指定下列标志:

   ```
   vmo -o v_pinshm=1
   ```

   有关如何运行 vmo 命令的详细指示信息，请参阅 AIX 手册。
2. （可选）如果您使用中等大小的页（这是缺省行为），那么请确保 DB2 实例所有者具有 CAP_BYPASS_RAC_VMM 和 CAP_PROPAGATE 功能。 例如:

   ```
   chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE db2inst1
   ```

   其中 *db2inst1* 是 DB2 实例所有者的用户标识。
3. 运行 bosboot 命令，以使 vmo 命令将在系统下一次引导之后生效。
4. 在服务器启动之后，使 DB2 数据库系统能够固定内存。
   a. 发出 db2set 命令，以将 **DB2_PINNED_BP** 注册表变量设置为 YES。
   b. 启动 DB2 数据库管理器。

   例如:

   ```
   db2set DB2_PINNED_BP=YES
   db2start
   ```

### 结果

完成这些步骤之后，DB2 数据库系统将指示操作系统固定 DB2 数据库共享内存。

# 第 2 章 多个 DB2 副本

对于版本 9 和更高版本，可以在同一台计算机上安装和运行多个 DB2 副本。DB2 副本指的是在同一台计算机上的特定位置安装的一个或多个 DB2 数据库产品。每个 DB2 版本 9 副本可以处于相同代码级别，也可以处于不同代码级别。

这样做的好处有:

- 能够同时在同一台计算机上运行需要不同 DB2 版本的应用程序
- 能够运行独立的 DB2 产品副本来实现不同的功能
- 在将生产数据库移至更新版本的 DB2 产品之前，能够在同一台计算机上测试
- 对于独立软件供应商，能够将 DB2 服务器产品嵌入到您的产品中，并对用户隐藏 DB2 数据库。对于 COM+ 应用程序，应对应用程序使用和分发 IBM 数据服务器 ODBC 和 CLI 驱动程序，而不是使用数据服务器运行时客户机，原因是一次只能对 COM+ 应用程序使用一个数据服务器运行时客户机。IBM 数据服务器 ODBC 和 CLI 驱动程序没有此限制。

## 缺省 IBM 数据库客户机接口副本

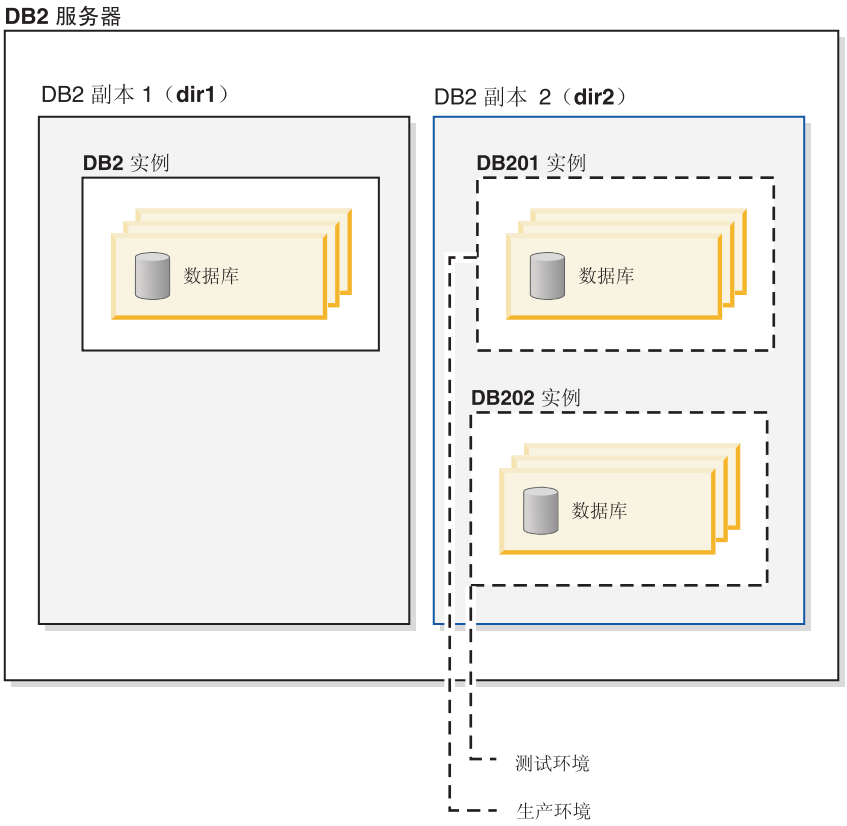
单台计算机上可以有多个 DB2 副本和一个缺省 IBM 数据库客户机接口副本，客户机应用程序通过该接口副本具有缺省情况下与数据库交互所需的 ODBC 驱动程序、CLI 驱动程序和 .NET 数据提供程序代码。

在版本 9.1（和更高版本）中，IBM 数据库客户机接口副本的代码随 DB2 副本一起提供。对于版本 9.5（和更高版本），可以选择安装一个新产品，它包含允许客户机应用程序与数据库交互所需的代码。此产品是 IBM 数据服务器驱动程序包（DSDRIVER）。对于版本 9.5（和更高版本），可以将 IBM 数据服务器驱动程序副本上的 DSDRIVER 安装在不同于安装 DB2 副本的位置。

在版本 9.1 后，可以在计算机上安装多个 DB2 副本；在版本 9.5 后，可以在计算机上安装多个 IBM 数据库客户机接口副本和多个 DB2 副本。在安装新的 DB2 副本或新的 IBM 数据服务器驱动程序副本期间，可以更改缺省 DB2 副本和缺省 IBM 数据库客户机接口副本。

下图显示了 DB2 服务器上安装的多个 DB2 副本，它们可以是 DB2 数据库产品的任意组合:

© Copyright IBM Corp. 1993, 2010




版本 8 和版本 9（或更高版本）副本可以在同一台计算机上共存，但版本 8 必须是缺省 DB2 和 IBM 数据库客户机接口副本。不能在安装期间将缺省 DB2 副本或缺省 IBM 数据库客户机接口副本从版本 8 副本更改为版本 9（或更高版本）副本，也不能在以后运行“切换缺省副本”命令 db2swtch，除非您首先迁移或卸载版本 8 副本。如果在系统上存在版本 8 时运行 db2swtch 命令，那么您将会接收到一条错误消息，指示由于在系统上找到版本 8，因此您不能更改缺省副本。

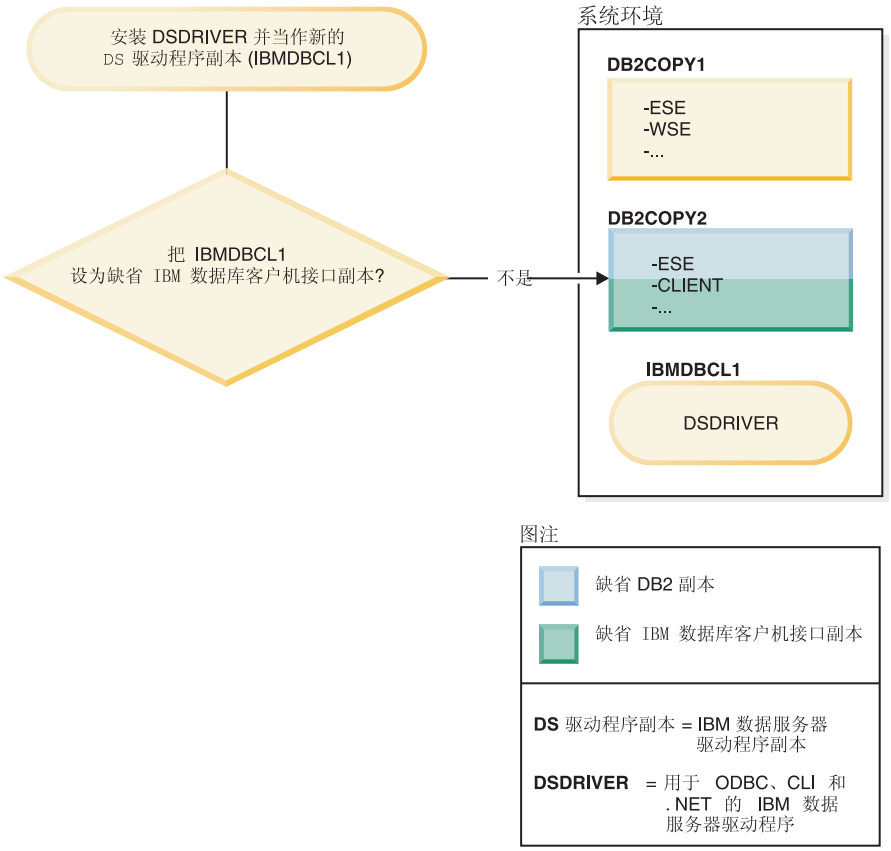
有时在安装多个 DB2 副本或多个 IBM 数据服务器驱动程序副本后，您可能要更改缺省 DB2 副本或缺省 IBM® 数据库客户机接口副本。如果安装了版本 8，那么需要先卸载它或者将它迁移到最低版本 9 才能更改缺省 DB2 副本，或者迁移到最低版本 9.5 才能更改缺省 IBM 数据库客户机接口副本。

客户机应用程序可以始终选择直接切换到数据服务器驱动程序位置，它是 DSDRIVER 的安装目录。

卸载作为缺省 IBM 数据库客户机接口副本的 DB2 副本或 IBM 数据服务器驱动程序副本时，将为您管理缺省副本。所选缺省副本将被除去，并为您选择新的缺省副本。卸载不是系统上的最后一个 DB2 副本的缺省 DB2 副本时，会要求您首先将缺省副本切换为另一个 DB2 副本。

## 安装新的 IBM 数据库客户机接口副本时选择缺省值

在版本 9.5 之后，考虑安装了两个 DB2 副本（DB2COPY1 和 DB2COPY2）的情况。DB2COPY2 是缺省 DB2 副本和缺省 IBM 数据库客户机接口副本。

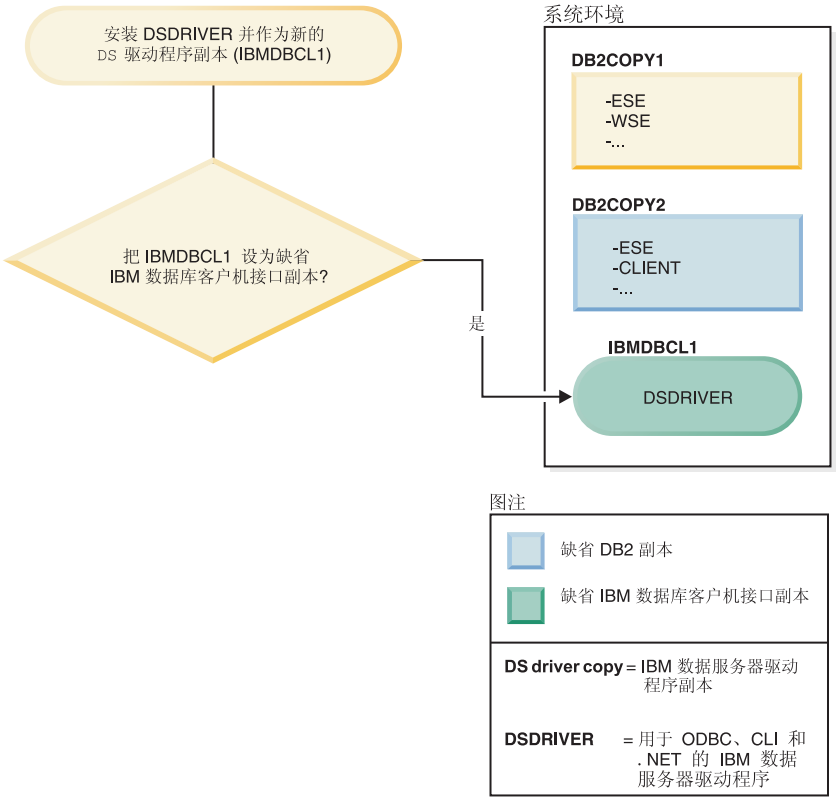



在新的 IBM 数据服务器驱动程序副本上安装 IBM 数据服务器驱动程序包（DSDRIVER）。

在安装新的 IBM 数据服务器驱动程序副本（IBMDBCL1）期间，将询问您是否要将新的 IBM 数据服务器驱动程序副本用作缺省 IBM 数据库客户机接口副本。

如果您回答“否”，那么 DB2COPY2 仍然是缺省 IBM 数据库客户机接口副本。（并且它将继续作为缺省 DB2 副本。）

但是，对于相同情况，如果您要将新的 IBM 数据服务器驱动程序副本用作缺省 IBM 数据库客户机接口副本，那么当系统询问时您应回答“是”。




在本例中，IBMDBCL1 成为缺省 IBM 数据库客户机接口副本。（DB2COPY2 仍然是缺省 DB2 副本。）

## 运行多个 DB2 副本时设置 DAS

从版本 9 开始，可以在同一台计算机上运行多个 DB2 副本。这将影响 DB2 管理服务器（DAS）的操作方式。DAS 是数据库管理器中的一个独特组件，它限于仅使一个版本处于活动状态，而无论在同一台计算机上安装了多少个 DB2 副本。为此，下列限制和功能需求适用。

在服务器上，只能有一个 DAS 版本并且它按如下所示管理实例:

- 如果 DAS 在版本 9.1 或版本 9.5 上，那么它可以管理版本 8 和版本 9.1 或版本 9.5 实例。
- 如果 DAS 在版本 8 上，那么它只能管理版本 8 实例。可以迁移版本 8 DAS 或删除它并创建新的版本 9.5 DAS，以便管理版本 8 和版本 9.1 实例。仅当要使用控制中心来管理实例时，才需要这样做。

无论在同一台计算机上安装了多少个 DB2 副本，任何时候在该计算机上都只能创建一个 DAS。此 DAS 将由同一台计算机上的所有 DB2 副本使用。在版本 9 或更高版本中，DAS 可属于当前安装的任何 DB2 副本。

要在一个版本 9.5 副本与另一个版本 9.5 副本之间移动 DAS，请使用 dasupdt 命令。要在版本 9.1 副本与版本 9.5 副本之间移动 DAS，不能使用 dasupdt，必须将版本 9.1 迁移至版本 9.5。

在 Windows 操作系统上，当需要将 DAS 移至相同版本的新缺省 DB2 副本时，也可以使用 dasupdt 命令。

**注**:

- dasupdt 命令只能用来在同一发行版的不同 DB2 副本之间（即，在不同的修订包之间）移动 DAS。它不能用来设置 DAS。
- 要从版本 8 迁移至版本 9.1，然后再迁移至版本 9.5 DAS，请使用 dasmigr 命令。
- 如果尚未设置 DAS，那么接下来应执行常规 DAS 设置过程，以在其中一个 DB2 副本上对其进行设置。

## 使用多个 DB2 副本时设置缺省实例（Windows）

从版本 9.1 起，就根据当前将环境设置为要使用的 DB2 副本来设置 DB2INSTANCE 环境。如果未将它显式设置为当前副本中的一个实例，那么会它缺省设置为使用 DB2INSTDEF 概要文件注册表变量指定的缺省实例。

DB2INSTDEF 是特定于当前正在使用的 DB2 副本的缺省实例变量。每个 DB2 副本都有它自己的 DB2INSTDEF 概要文件注册表变量。实例名称在系统上必须是唯一的；创建了某个实例之后，数据库管理器就会对现有副本进行扫描以确保该实例是唯一的。

使用多个 DB2 副本时，请遵循下列准则来设置缺省实例:

- 如果没有对特定 DB2 副本设置 DB2INSTANCE，那么将 DB2INSTDEF 的值用于该 DB2 副本。这意味着:
  - 如果 DB2INSTANCE=ABC 并且 DB2INSTDEF=XYZ，那么将使用的值为 ABC
  - 如果*未设置* DB2INSTANCE 并且 DB2INSTDEF=XYZ，那么将使用 XYZ
  - 如果*未设置* DB2INSTANCE 并且也*未设置* DB2INSTDEF，那么任何依赖于有效 DB2INSTANCE 的应用程序或命令都将无法工作。
- 可以使用 db2envar.bat 命令或 db2SelectDB2Copy API 来切换 DB2 副本。也可以适当地设置所有环境变量（例如，PATH、INCLUDE、LIB 和 DB2INSTANCE），但是您需要确保正确设置它们。

  **注**: 使用 db2envar.bat 命令与设置环境变量有些不同。db2envar.bat 命令确定它所属的 DB2 副本，然后将此 DB2 副本的路径添加到 PATH 环境变量的前面。

  当同一台机器上有多个 DB2 副本时，PATH 环境变量只能指向其中一个副本: 缺省副本。例如，如果 DB2COPY1 位于 c:\sqllib\bin 下并且是缺省副本，而 DB2COPY2 位于 d:\sqllib\bin 下，那么如果您想在常规命令窗口中使用 DB2COPY2，就应该在此命令窗口中运行 d:\sqllib\bin\db2envar.bat。这将调整此命令窗口的 PATH（和其他某些环境变量），以便它将选取 d:\sqllib\bin 中的二进制文件。

- DB2INSTANCE 仅对正在使用的 DB2 副本中的实例有效。但是，如果通过运行 db2envar.bat 命令来切换副本，那么 DB2INSTANCE 将更新为最初切换至的 DB2 副本的 DB2INSTDEF 值。
- DB2INSTANCE 是正在该 DB2 副本中执行的应用程序将使用的当前 DB2 实例。缺省情况下，当您在副本之间切换时，DB2INSTANCE 将更改为该副本的 DB2INSTDEF 的值。由于所有实例都在当前副本中，因此 DB2INSTDEF 在一个副本系统中的意义不是很大；但是，如果未设置另一个实例，仍然可以将 DB2INSTDEF 作为缺省实例。
- 除非您使用 SET VARIABLE=<variable_name> 指定全局概要文件注册表变量，否则所有全局概要文件注册表变量都特定于 DB2 副本。

## 数据库管理器的多个实例

可以在一台服务器上创建多个数据库管理器实例。这意味着可以在一台物理计算机上创建同一个产品的几个实例，并使它们同时运行。这在设置环境方面提供了灵活性。

**注:** 在两个不同的 DB2 副本中不能使用相同的实例名。

您可能希望有多个实例来创建下列环境:

- 将开发环境与生产环境分离。
- 针对环境要服务的特定应用程序单独调整每一个环境。
- 保护敏感信息，使管理员无法对其进行访问。例如，可能希望将工资单数据库保护在它自己的实例中，以使其他实例的所有者不能查看工资单数据。

**注:**

- （仅在 UNIX® 操作系统上）：要防止两个或多个实例之间的环境冲突，应确保每个实例主目录位于本地文件系统上。
- （仅在 Windows® 平台上）：在节点目录中将实例编目为本地的或远程的。缺省实例由 DB2INSTANCE 环境变量来定义。可以与其他实例连接（ATTACH），以便执行只能在实例级执行的维护和实用程序任务，如创建数据库、强制断开应用程序、监视数据库或更新数据库管理器配置。当试图与不在缺省实例中的实例连接时，将使用该节点目录来确定如何与该实例通信。
- （在任何平台上）：DB2 数据库程序文件以物理方式存储在一个位置，并且每个实例都指向该实例所属的副本，这样就不必为创建的每个实例复制程序文件。几个相关的数据库可以位于单个实例内。

## 多个实例（Windows）

可以在同一台计算机上运行 DB2 数据库管理器的多个实例。数据库管理器的每个实例维护其自己的数据库且具有自己的数据库管理器配置参数。

**注:** 实例也可以属于计算机上处于不同数据库管理器级别的不同 DB2 副本。如果您正在运行 64 位 Windows 系统，那么可安装 32 位 DB2 或 64 位 DB2，但这两者不能在同一机器上共存。

数据库管理器实例由下列内容组成:

- 表示该实例的 Windows 服务。服务的名称与实例名相同。服务的显示名称（在**服务**面板中）是实例名加上“DB2 -”字符串前缀。例如，对于名为“DB2”的实例，存在名为“DB2”并且显示名为“DB2 - *DB2 副本名* - DB2”的 Windows 服务。

**注**: 不会为客户机实例创建 Windows 服务。

- 实例目录。此目录包含数据库管理器配置文件、系统数据库目录、节点目录、数据库连接服务（DCS）目录以及与实例相关联的所有诊断日志和转储文件。实例目录随 Windows 操作系统系列的版本不同而不同；要验证 Windows 上的缺省目录，请使用 db2set DB2INSTPROF 命令来检查 **DB2INSTPROF** 环境变量的设置。您还可以通过更改 **DB2INSTPROF** 环境变量来更改缺省实例目录。例如，将其设置为 c:\DB2PROFS:
  - 使用 db2set.exe **-g** 命令将 **DB2INSTPROF** 设置为 c:\DB2PROFS
  - 运行 DB2ICRT.exe 命令来创建此实例。
- 在 Windows 操作系统上创建实例时，用户数据文件（例如，实例目录和 db2cli.ini 文件）的缺省位置为下列目录:
  - 在 Windows XP 和 Windows 2003 操作系统上: Documents and Settings\All Users\Application Data\IBM\DB2*Copy Name*
  - 在 Windows 2008 和 Windows Vista（和更高版本）操作系统上: Program Data\IBM\DB2*Copy Name*

  其中，*Copy Name* 表示 DB2 副本名。

**注**: db2cli.ini 文件的位置可能会根据是否使用了 Microsoft® ODBC 驱动程序管理器、所使用的数据源名称（DSN）的类型、所安装的客户机或驱动程序的类型以及是否设置了注册表变量 **DB2CLIINIPATH** 而不同。有关更多信息，请参阅 *Call Level Interface Guide and Reference, Volume 1* 中的“db2cli.ini initialization file”主题。

## 更新 DB2 副本（Windows）

更新 DB2 产品时，将需要指定是更新现有 DB2 副本，还是安装一个新的副本。要更新 DB2 副本，必须选择使用现有产品选项。

不能同时更新多个 DB2 副本。为了更新同一台计算机上安装的其他 DB2 副本，需要重新运行安装。安装过程使您能够选择是迁移版本 8 或版本 9.1 副本（在同一路径中）还是安装新的版本 9.1 或版本 9.5 副本而不必修改版本 8 安装。

- 如果选择迁移，那么将除去版本 8 安装。
- 如果选择安装新的 DB2 副本，那么稍后可以选择使用 db2ckmig 和 db2imigr 命令迁移实例。

可以使用 db2iupdt 命令在版本 9.1 或版本 9.5 的不同 DB2 副本之间移动 DB2 实例，并使用 db2imigr 命令将版本 8 实例移至版本 9.1 或版本 9.5。

**注**:

- 不支持版本 7 和版本 9.1 或版本 9.5 共存。
- 不支持 32 位 DB2 数据服务器和 64 位 DB2 数据服务器在同一台 Windows X64 计算机上共存。

  无法从版本 8 的 32 位 X64 DB2 安装迁移至版本 9.1 或版本 9.5 的 64 位安装。相反，需要迁移至版本 9.1 或版本 9.5 的 32 位以使用 X64 DB2 数据服务器安装来移至 64 位。32 位版本将被除去。如果安装了多个 32 位 DB2 副本，那么需要将所有实例移至一个 DB2 副本并从计算机中除去这些副本。

- 如果有多个版本 9.1 或版本 9.5 副本，那么安装选项为“安装新副本”或“使用现有 DB2 副本”，可以升级或添加新功能部件。如果您不仅具有版本 9.1 或版本 9.5 副本，还具有版本 8 副本，那么迁移选项将可用。
- 如果安装了版本 8 或版本 9.1，那么安装选项为“将现有版本 8 或版本 9.1 迁移至版本 9.5 副本”或“安装新 DB2 副本”。
- 如果安装了版本 7 或更早版本，那么安装将显示一条消息，指示不支持迁移至版本 9.1 或版本 9.5。只有在卸载了版本 7 之后才能安装新的 DB2 副本。也就是说，版本 7 和版本 9.1 或版本 9.5 不能共存。
- 要将实例从一个版本 9.1 或版本 9.5 副本移至另一个副本，可以使用 db2iupdt 命令。
- 如果使用 db2imigr 命令从版本 8 迁移实例，将需要重新配置任何 ODBC 数据源。

## 同时运行多个实例（Windows）

可以在同一 DB2 副本或不同 DB2 副本中同时运行多个实例。

要使用命令行在同一 DB2 副本中同时运行多个实例:

1. 输入以下命令，将 DB2INSTANCE 变量设置为要启动的另一个实例的名称:

   ```
   set db2instance=<another_instName>
   ```

2. 通过输入 db2start 命令来启动实例。

要在不同 DB2 副本中同时运行多个实例，请使用下列任一方法:

- 通过以下途径使用 DB2 命令窗口：选择“开始”→“程序”→“IBM DB2”→ *<DB2 副本名称>* →“命令行工具”→“DB2 命令窗口”。已使用选择的特定 DB2 副本的正确环境变量设置该命令窗口。
- 从命令窗口中使用 db2envar.bat:
  1. 打开命令窗口。
  2. 通过使用想要应用程序使用的 DB2 副本的标准路径来运行 db2envar.bat 文件:

     ```
     <DB2 Copy install dir>\bin\db2envar.bat
     ```

在切换至特定 DB2 副本后，使用上面部分“要在同一 DB2 副本中同时运行多个实例”中指定的方法来启动实例。

## 使用同一 DB2 副本或不同 DB2 副本上的实例

可以在同一 DB2 副本或不同 DB2 副本中同时运行多个实例。

要使用同一 DB2 副本中的多个实例，您需要:

1. 创建所有实例或将它们迁移至同一 DB2 副本。
2. 在对要使用的实例发出命令之前，将 DB2INSTANCE 环境变量设置为该实例的名称。

要阻止实例访问另一实例的数据库，可在与实例同名的目录下为实例创建数据库文件。例如，在驱动器 C: 上为实例“DB2”上创建数据库时，会在称为 C:\DB2 的目录中创建数据库文件。类似地，在驱动器 C: 上为实例 TEST 创建数据库时，会在称为 C:\TEST 的目录中创建数据库文件。缺省情况下，它的值为安装了 DB2 产品的盘符。有关更多信息，请参阅 *dftdbpath* 数据库管理器配置参数。

要在具有多个 DB2 副本的系统中使用实例，请使用下列任一方法:

- 通过以下途径使用命令窗口：选择“开始”→“程序”→“IBM DB2”→ *<DB2 副本名称>* →“命令行工具”→“命令窗口”。已使用选择的特定 DB2 副本的正确环境变量设置该命令窗口。
- 从命令窗口中使用 db2envar.bat:
  1. 打开命令窗口。
  2. 通过使用想要应用程序使用的 DB2 副本的标准路径来运行 db2envar.bat 文件:

     ```
     <DB2 Copy install dir>\bin\db2envar.bat
     ```

# 第 3 章 自主计算

DB2 自主计算环境能够自我配置、修复、优化和保护。自主计算通过对发生的各种情况进行检测和作出响应，将由数据库管理员来管理计算环境改变为通过一些技术来管理。

下列自动功能可帮助您管理数据库系统:

- 自调整内存
- 自动存储器
- 自动创建（压缩）字典（ADC）
- 自动数据库备份
- 自动收集统计信息
- 配置顾问程序
- 运行状况监视器
- 实用程序调速

## 自动功能

自动功能可帮助您管理数据库系统。它们使得系统能够执行自诊断，并通过针对历史问题数据来分析实时数据，从而预测可能会发生的问题。可以配置一些自动工具以在无外部干预的情况下更改系统，从而避免服务中断。

创建数据库时，缺省情况下会启用下列某些自动功能，但是其他自动功能需要您手动启用:

**自调整内存（仅适用于单一分区数据库）**

自调整内存功能简化了内存配置任务。此功能通过反复地自动调整某些内存配置参数的值和缓冲池大小来对工作负载的显著变化作出响应，从而优化性能。内存调整器会在多个内存使用者（包括排序功能、程序包高速缓存、锁定列表和缓冲池）之间动态地分配可用内存资源。在创建数据库之后，可以通过将数据库配置参数 **self_tuning_mem** 设置为 OFF 来禁止对内存进行自调整。

**自动存储器**

自动存储器功能简化了表空间的存储管理。创建数据库时，可以指定数据库管理器将用来存放表空间数据的存储路径。然后，当您创建并填充表空间时，数据库管理器将管理这些表空间的容器和空间分配。

**自动对行进行压缩**

可以对表进行压缩以节省存储空间。将完全自动进行压缩；一旦您指定应使用 CREATE TABLE 或 ALTER TABLE 语句的 **COMPRESS YES** 子句对表进行压缩，就不需要再执行任何操作来管理压缩。（将未压缩的现有表或索引转换为压缩的表或索引时，需要执行 REORG 才能压缩现有数据）。

**自动数据库备份**

数据库可能会由于各种硬件或软件故障而变得不可用。确保有最新的完全数据库备份是规划和实现系统灾难恢复策略的主要部分。通过在灾难恢复策略中使用自动数据库备份功能，数据库管理器就能够正确并且定期地备份数据库。

© Copyright IBM Corp. 1993, 2010

**自动收集统计信息**

自动收集统计信息通过确保您具有最新的表统计信息来改善数据库性能。数据库管理器确定工作负载需要哪些统计信息以及需要更新哪些统计信息。通过在编译 SQL 语句时收集运行时统计信息，可以用异步（在后台中）或同步方式收集统计信息。然后，DB2 优化器根据准确的统计信息来选择访问方案。在创建数据库之后，可以通过将数据库配置参数 **auto_runstats** 设置为 OFF 来禁用自动收集统计信息。仅当启用了自动收集统计信息时，才能启用收集实时统计信息。收集实时统计信息由 **auto_stmt_stats** 配置参数控制。

**配置顾问程序**

创建数据库时，将自动运行此工具来确定并设置数据库配置参数和缺省缓冲池（IBMDEFAULTBP）的大小。根据系统资源和系统的用途选择值。此初始自动调整意味着您的数据库比使用缺省值创建的数据库具有更好的性能。它还意味着在创建数据库之后您将花费较少时间来调整系统。任何时候（即使在填充了数据库之后）都可以运行“配置顾问程序”，以让工具根据当前系统特征来建议一组配置参数并且可以选择应用这些参数来优化性能。

**运行状况监视器**

运行状况监视器是一个服务器端工具，它主动监视数据库环境中可能导致性能下降或潜在中断的情况或变动。不需要您进行任何形式的监视活动就可以产生一些运行状况信息。如果运行状况不正常，数据库管理器就会通知您并建议您如何继续执行操作。运行状况监视器使用快照监视器来收集关于系统的信息，不会造成性能损失。此外，它不打开任何快照监视开关来收集信息。

**实用程序调速**

此功能调整各种维护实用程序对性能的影响，以便在生产期间可以同时运行这些维护实用程序。虽然缺省情况下定义了已调速实用程序的*影响策略*，但是如果您想运行已调速实用程序，那么必须设置*影响优先级*。调速系统确保已调速实用程序尽可能频繁地运行而不违反影响策略。目前，可以调速统计信息收集、备份操作、重新平衡操作和异步索引清除。

## 自动维护

数据库管理器提供了自动维护功能，即执行数据库备份、保持统计信息是最新的以及在必要时重组表和索引。对于确保数据库具有最佳性能和可恢复性来说，对数据库执行维护活动十分必要。

维护数据库时将执行下面的某些或所有活动:

- **备份。**备份数据库时，数据库管理器将复制数据库中的数据并将它们存储在另一介质上，以防原始介质发生故障或毁坏。自动进行数据库备份有助于确保定期正常地备份数据库，从而使您不必担心何时进行备份，也不需要了解 BACKUP 命令的语法。
- **数据碎片整理（表或索引重组）。**此维护活动可以提高数据库管理器访问表的效率。自动重组功能负责管理脱机进行的表和索引重组，从而使您不必担心何时以及如何重组数据。
- **数据访问优化（统计信息收集）。**数据库管理器将更新有关表数据、索引数据或者表数据及其索引数据的系统目录统计信息。优化器使用这些统计信息来确定用来访问数据的路径。自动收集统计信息功能通过维护最新的表统计信息来尝试提高数据库的性能。目标是允许优化器根据准确的统计信息来选择访问方案。

- **统计信息概要分析。**自动统计信息概要分析功能通过执行下列操作来建议何时以及如何收集表统计信息：检测过时的、丢失的或不正确的统计信息，以及根据查询反馈信息来生成统计概要文件。

确定是否运行以及何时运行维护活动可能相当费时，但使用自动维护功能就可以为您解除此负担。可以使用自动维护数据库配置参数来简单灵活地管理自动维护功能的启用。通过使用“配置自动维护”向导，可以指定维护目标。数据库管理器使用这些目标来确定是否需要执行维护活动，并且在下一个可用的维护时间段（由您定义的时间段）仅运行必需的维护活动。

## 维护时间段

维护时间段是您定义的用于运行自动维护活动的时间段，自动维护活动包括备份、统计信息收集、统计信息概要分析和重组。脱机时间段可能就是无法访问数据库的时间段。联机时间段可能就是允许用户连接至数据库的时间段。

维护时间段不同于任务时间表。在维护时间段内，不必运行每项自动维护活动。数据库管理器会对系统进行评估以确定是否需要运行每项维护活动。如果未满足维护需求，就运行该维护活动。如果数据库的维护状态良好，那么不运行维护活动。

确定您希望何时运行自动维护活动。运行自动维护活动将消耗系统上的资源，并且还可能会影响数据库的性能。其中某些活动还会限制访问表、索引和数据库。因此，必须提供数据库管理器可以运行维护活动的适当时间段。使用控制中心或运行状况中心中的“自动维护”向导将这些时间段指定为脱机和联机维护时间段。

**脱机维护活动**

脱机维护活动（即，脱机数据库备份以及表和索引重组）是只能在脱机维护时间段内进行的维护活动。影响用户访问的程度取决于所运行的维护活动:

- 在脱机备份期间，没有任何应用程序可以与数据库连接。当前已连接的所有应用程序都会被强制断开连接。
- 在进行脱机表或索引重组（数据碎片整理）期间，应用程序可以访问但是不能更新表中的数据。

即使超过了指定的时间段，脱机维护活动也将运行直到完成为止。经过一段时间之后，内部调度机制将了解如何最佳估计作业完成时间。如果脱机维护时间段对于特定数据库备份或重组活动来说太短了，那么调度程序下一次将不启动作业，并依赖运行状况监视器来提供需要延长脱机维护时间段的通知。

**联机维护活动**

联机维护活动（即，自动统计信息收集和概要分析、联机索引重组以及联机数据库备份）是只能在联机维护时间段内进行的维护活动。运行联机维护活动时，允许任何当前已连接的应用程序保持连接，并允许建立新连接。为了使它们对系统产生的影响最小，将用适当的实用程序调节机制来调节联机数据库备份以及自动统计信息收集和概要分析。

即使超过了指定的时间段，联机维护活动也将运行直到完成为止。

# 自调整内存

从 DB2 V9 开始，新的内存调整功能会通过自动设置一些内存配置参数的值来简化内存配置任务。启用此功能之后，内存调整器就会在下列内存使用者之间动态分配可用内存资源：缓冲池、程序包高速缓存、锁定内存和排序内存。

调整器在 **database_memory** 配置参数定义的内存限制范围内工作。**database_memory** 值本身也可以自动调整。对 **database_memory** 启用自动调整（将它设置为 AUTOMATIC）之后，调整器将确定数据库的整体内存需求并根据当前数据库需求来增加或减少分配给数据库共享内存的内存量。例如，如果当前数据库需求很高，并且系统上有足够的可用内存，那么数据库共享内存将消耗较多的内存。如果数据库内存需求下降，或者系统上的可用内存量变得过低，就会释放一些数据库共享内存。

如果您未对自调整启用 **database_memory** 参数（未将此参数设置为 AUTOMATIC），那么整个数据库都将使用您为此参数指定的内存量，从而根据需要在数据库内存使用者之间分配内存。可以通过两种方法指定数据库使用的内存量：将 **database_memory** 设置为一个数值或者将它设置为 COMPUTED。在第二种情况下，总内存量是根据数据库启动时的数据库内存堆初始值总计而计算的。

除了使用 **database_memory** 配置参数来调整数据库共享内存以外，您还可以使其他内存使用者进行自调整，如下所示:

- 对于缓冲池，使用 ALTER BUFFERPOOL 和 CREATE BUFFERPOOL 语句。
- 对于程序包高速缓存，使用 **pckcachesz** 配置参数。
- 对于锁定内存，使用 **locklist** 和 **maxlocks** 配置参数。
- 对于排序内存，使用 **sheapthres_shr** 和 **sortheap** 配置参数。

## DB2 中的内存分配

在 DB2 中，内存分配和释放在各个时间进行。当发生指定事件（例如，应用程序连接时）时可以将内存分配给特定内存区，或者可以根据配置参数设置中的更改将其释放。

下图显示了数据库管理器为不同用途分配的各个内存区和允许控制此内存大小的配置参数。请注意，在包括多个逻辑数据库分区的企业服务器版环境中，每个数据库分区都有它自己的“数据库管理器共享内存”集。

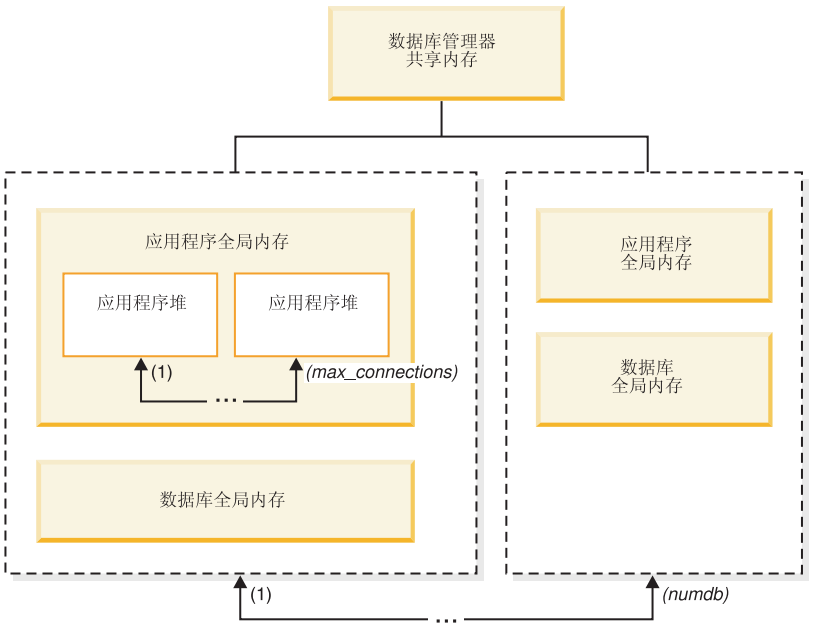



*图 1. 数据库管理器所用的内存类型*

在发生下列事件时，将为数据库管理器的每个实例分配内存:

- **当启动数据库管理器（db2start）时**: 分配数据库管理器全局共享内存（也称为实例共享内存），并且会一直保持此分配状态，直到数据库管理器停止运行（db2stop）为止。此区域中包含数据库管理器在管理所有数据库连接上的活动时所需的信息。DB2 将自动控制数据库管理器全局共享内存大小。
- **当第一次激活数据库或与数据库连接时**: 将分配数据库全局内存。所有与数据库连接的应用程序均使用数据库全局内存。数据库全局内存的大小由 **database_memory** 配置参数指定。缺省情况下，此参数设置为 automatic，允许 DB2 计算为数据库分配的初始内存量，并在运行时根据数据库的需要来自动配置数据库内存大小。可以设置 **database_memory** 来分配大于最初所需的内存，以便以后可以动态地分配附加内存。

  可以动态地调整下列内存区，例如，减少分配给一个内存区的内存和增大另一个内存区中的内存。

  – 缓冲池（使用 ALTER BUFFERPOOL DDL 语句）
  – 数据库堆（包括日志缓冲区）
  – 实用程序堆
  – 程序包高速缓存
  – 目录高速缓存
  – 锁定列表

  在已启用数据库管理器分区内并行性配置参数（intra_parallel）的环境、已启用连接集中器的环境或者已启用数据库分区功能（DPF）的环境中，还可以分配共享排序

堆作为数据库全局内存的一部分。另外，如果数据库管理器配置参数 **sheapthres** 设置为 0（缺省值），那么所有排序都将使用数据库全局内存。

- **当应用程序连接至数据库时**：分配了应用程序堆。每个应用程序都具有它自己的应用程序堆。如果需要，可以使用 **applheapsz** 配置参数来限制任何一个应用程序可以分配的内存量，或者使用 **appl_memory** 配置参数来限制消耗的应用程序内存总量。

  数据库管理器配置参数 **max_connections** 设置可以与实例连接或者与该实例中存在的任何数据库连接的应用程序数的上限。因为每个与数据库连接的应用程序都涉及到分配一些内存，所以允许大量并发应用程序可能将使用更多内存。
- **创建代理程序的时间**：当出现连接请求或并行环境中出现新的 SQL 请求时，系统会指定一个代理程序并为其分配代理程序专用内存。为该代理程序分配代理程序专用内存，其中包含仅用于该特定代理程序的内存，例如，专用排序堆。

此外，图中还指定了下列配置参数设置，它们用于限制为每种类型的内存区分配的内存大小。请注意，在分区数据库环境中，将在每个数据库分区中分配此类内存。

- **numdb**

  此参数指定不同应用程序可以使用的并发活动数据库的最大数目。由于每个数据库都有其自身的全局内存区，因此在增大此参数的值时，所分配的内存容量也可能会增加。
- **maxappls**

  此参数指定一个数据库可以同时连接的应用程序的最大数目。它将影响可能为该数据库分配的代理程序专用内存容量和应用程序全局内存容量。请注意，对于每个数据库，可将此参数设置为不同的值。

  另外两个需要考虑的参数是 **max_coordagents** 和 **max_connections**，它们都在实例级别应用（对 DPF 实例上的每个节点）。
- **max_connections**

  此参数用于限制任何时候可访问 DB2 服务器的连接数或实例连接数（对 DPF 实例上的每个节点）。
- **max_coordagents**

  此参数用于限制一个实例中的所有活动数据库中可以同时存在的数据库管理器协调代理程序数（对 DPF 实例上的每个节点）。与 **maxappls** 和 **max_connections** 一起，此参数可以限制为代理程序专用内存和应用程序全局内存分配的内存容量。

db2mtrk 命令调用的内存跟踪程序允许您查看实例中的当前内存分配，包括有关每个内存池的下列类型的信息：

- 当前大小
- 最大大小（硬性限制）
- 最大大小（高水位标记）

在 Unix 和 Linux 上，尽管可以使用 ipcs 命令来列示所有共享内存段，但是它并不能准确反映所耗用的资源数量。可以使用 db2mtrk 命令来替代 ipcs。

# 自调整内存操作的详细信息和局限性

## 确定调整需求

为了确保在两个内存使用者之间进行公平比较，开发了一种新的公共度量。每个已调整的内存使用者根据附加内存计算预测好处，并将它报告给自调整内存进程。自调整内存使用这些数字作为内存调整的基础，从最不需要的使用者收回内存并将它重新分配给将获得最多好处的那些内存区域。

## 内存调整的频率

以 3 分钟作为时间间隔来执行内存调整。根据在此时间间隔内测量的性能数据和其他条件（例如，内存可用性）来作出内存调整决定。

## 跟踪自调整内存的进度

可以使用 GET DATABASE CONFIGURATION 命令或使用快照来获取当前内存配置。自调整所作的更改将记录在 stmmlog 目录中的内存调整日志文件中。内存调整日志文件包含每个内存使用者在每个调整时间间隔内的资源需求摘要。可以根据日志条目中的时间戳记确定这些时间间隔。

## 获得最佳配置期望所用的时间

使此功能处于已启用状态会导致快速调整参数来优化内存使用情况。最少只要 1 个小时就可以通过初始配置调整系统。大多数情况下，通常最多 10 个小时就完成调整。当针对数据库运行的查询明显展示不同的内存特征时，就会出现这种最坏的情况。

## 自调整内存的局限性

在少量内存可用的情况下（例如，因为 *database_memory* 的值设置得很小，或者因为多个数据库、实例或其他应用程序正在服务器上运行），自调整内存带来的性能好处将有限。

因为自调整内存根据数据库工作负载作出调整决定，所以内存特征不断变化的工作负载将限制自调整内存的能力以有效地进行调整。如果工作负载的内存特征不断变化，那么自调整内存调整内存的频率将会降低，并且将重复调整以改变目标条件。在这种情况下，自调整内存将无法获得绝对汇合，而是尝试维护调整为当前工作负载的内存配置。

# 内存参数交互和局限性

尽管您可以启用自调整内存并对大多数与内存相关的配置参数使用缺省 AUTOMATIC 设置，但是知道不同内存参数的局限性以及它们之间的交互将非常有用，这样就可以更好地控制它们的设置，并了解在某些情况下仍然可能发生“内存不足”错误的原因。

## 内存类型

基本上 DB2 数据库管理器使用两种类型的内存:

**性能内存**

这是用来提高数据库性能的内存。性能内存由自调整内存管理器 (STMM) 控制

并分发给各种性能堆。您可以将 **database_memory** 配置参数设置为性能内存的最大容量，也可以将 **database_memory** 设置为 AUTOMATIC 以便让 STMM 管理性能内存的全部容量。

**功能内存**

此内存由应用程序使用。您可以使用 **appl_memory** 配置参数，来控制 DB2 数据库代理程序为了对应用程序请求提供服务而分配的功能内存（或应用程序内存）的最大容量。缺省情况下，此参数设置为 AUTOMATIC，这意味着只要有系统资源可用，就允许功能内存请求。如果您要使用具有内存使用量限制的 DB2 数据库产品，或者您将 **instance_memory** 设置为特定值，那么在数据库分区所分配的内存总量未超过 **instance_memory** 限制的情况下，将强制使用 **instance_memory** 限制并且允许功能内存请求。

在 AUTOMATIC 设置可用之前，可以使用各种操作系统和 DB2 工具来查看不同类型的内存（例如，共享内存、专用内存、缓冲池内存、锁定列表、排序内存（堆）等等）所耗用的空间量，但几乎不可能查看 DB2 数据库管理器耗用的内存总量。如果其中一个堆达到内存限制，那么应用程序中的某个语句将失败，并且将显示“内存不足”错误消息。即使增大该堆的内存量并重新运行应用程序，对另一个堆执行另一个语句时也可能会发生“内存不足”错误。现在，可以使用缺省 AUTOMATIC 配置参数设置来除去各个功能内存堆的硬上限。

有需要时（例如，要避免出现数据库应用程序性能低下的情况，需要极其大量的内存），可使用 **appl_memory** 配置参数在数据库级别对整体应用程序内存应用限制。另外，还可以将该堆的相应数据库配置参数由 AUTOMATIC 设置更改为固定值，从而对各个堆应用限制。如果所有功能内存堆的所有配置参数都设置为 AUTOMATIC，并且强制施加了 **instance_memory** 限制，那么对应用程序内存耗用量的唯一限制是 **instance_memory** 限制。如果您还将 **instance_memory** 设置为 AUTOMATIC，并且您要使用具有内存使用量限制的 DB2 数据库产品，那么 DB2 数据库管理器将自动确定内存耗用量上限。

通过使用 db2pd -dbptnmem 命令或 ADMIN_GET_DBP_MEM_USAGE 表函数，您可以非常方便地查看实例内存耗用总量以及当前 **instance_memory** 耗用量。

## 内存配置参数之间的交互

当自调整内存管理器 (STMM) 处于活动状态并且启用了数据库内存的自调整时（**database_memory** 设置为 AUTOMATIC），那么 STMM 将检查系统上的可用内存量，并自动确定为了获取最佳性能而应该供性能堆专用的内存量。所有性能堆都将计入 **database_memory** 总大小。除了性能内存需求以外，还需要一定内存来确保 DB2 数据库管理器的操作和完整性。**instance_memory** 耗用的空间量与这两个内存使用者所需空间量之差可供应用程序内存 (**appl_memory**) 使用。将根据需要来分配应用程序的功能内存。如果未强制施加 **instance_memory** 限制，那么对单个应用程序可分配的内存量没有其他限制。

如果存在 **instance_memory** 限制，根据配置，STMM 还会定期查询剩余的可用系统内存量以及剩余的可用 **instance_memory** 空间量。为了防止应用程序发生故障，STMM 认为应用程序需求优先于性能条件。有需要时，它将通过减少可供性能堆使用的空间量使性能下降，从而提供足够的可用系统内存和 **instance_memory** 空间以满足应用程序内存请求。应用程序完成时，使用的内存将被释放，从而可供其他应用程序重复使用或者为 **database_memory** 回收以供 STMM 使用。如果数据库系统的性能在

应用程序活动频繁期间变得不可接受，那么最好对允许数据库管理器运行的应用程序数目进行控制（例如，使用连接集中器或者 DB2 版本 9.5 新增的工作负载管理功能部件），或者考虑对系统添加内存资源。

## 启用自调整内存功能

自调整内存功能通过自动设置内存配置参数值以及调整缓冲池大小来简化内存配置任务。

### 关于本任务

启用此功能后，内存调整器就会在几个内存使用者（包括排序、程序包高速缓存、锁定列表区域和缓冲池）之间动态地分配可用内存资源。

### 过程

1. 通过将 **self_tuning_mem** 设置为 ON 来对数据库启用自调整功能。可以使用 UPDATE DATABASE CONFIGURATION 命令、sqlfupd API 或通过控制中心中的更改数据库配置参数窗口来将 **self_tuning_mem** 设置为 ON。
2. 要对由内存配置参数控制的内存区域启用自调整功能，请使用 UPDATE DATABASE CONFIGURATION 命令、sqlfupd API 或通过控制中心中的更改数据库配置参数窗口将相关配置参数设置为 AUTOMATIC。
3. 要对缓冲池启用自调整功能，请将缓冲池大小设置为 AUTOMATIC。可以使用 ALTER BUFFER POOL 语句（对于现有缓冲池）或 CREATE BUFFER POOL 语句（对于新缓冲池）来完成此操作。如果在分区数据库环境中将缓冲池大小设置为 AUTOMATIC，那么就不应该在 SYSCAT.BUFFERPOOLDBPARTITIONS 中为该缓冲池定义任何条目。

### 结果

**注:**

1. 由于自调整功能在不同内存区域之间重新分配内存，所以，必须至少启用两个内存区域（例如锁定内存区域和数据库共享内存区域）才能使自调整功能起作用。唯一的例外是由 **sortheap** 配置参数控制的内存。当仅将 **sortheap** 设置为 AUTOMATIC，**sheapthres** 设置为 0 时，将启用 **sortheap** 的自调整功能。
2. 要为自调整功能启用 **locklist** 配置参数，还必须为自调整功能启用 **maxlocks**，因此当 **locklist** 设置为 AUTOMATIC 时，**maxlocks** 也设置为 AUTOMATIC。要为自调整功能启用 **sheapthres_shr** 配置参数，还必须为自调整功能启用 **sortheap**，因此当 **sheapthres_shr** 设置为 AUTOMATIC 时，**sortheap** 也设置为 AUTOMATIC。
3. 仅当数据库管理器配置参数 **sheapthres** 设置为 0 时，才允许自动调整 **sheapthres_shr** 或 **sortheap**。
4. 自调整内存功能仅在 HADR 主服务器上运行。在 HADR 系统上激活自调整内存功能后，永远不会在辅助服务器上运行此功能，并且，仅当正确地设置配置后，此功能才会在主服务器上运行。如果运行了切换 HADR 数据库角色的命令，自调整内存操作也会切换，从而在新的主服务器上运行。在通过接管操作启动主数据库或将备用数据库转换为主数据库之后，直到第一个客户机连接生效，STMM EDU 才可能启动。

## 禁用自调整内存功能

可以通过将 **self_tuning_mem** 设置为 OFF 来对整个数据库禁用自调整功能。当 **self_tuning_mem** 设置为 OFF 时，设置为 AUTOMATIC 的内存配置参数和缓冲池仍为 AUTOMATIC，并且内存区保持其当前大小不变。

可以使用 UPDATE DATABASE CONFIGURATION 命令、sqlfupd API 或通过控制中心中的更改数据库配置参数窗口来将 **self_tuning_mem** 设置为 OFF。

如果只有一个内存使用者启用了自调整功能，那么还可以有效地对整个数据库取消激活自调整功能。这是因为，当仅启用了一个内存区时，不能对内存进行再分布。

例如，要禁用 **sortheap** 配置参数自调整功能，可以输入以下语句:

```
UPDATE DATABASE CONFIGURATION USING SORTHEAP MANUAL
```

要禁用 **sortheap** 配置参数自调整功能，并同时将 **sortheap** 的当前值更改为 2000，请输入以下语句:

```
UPDATE DATABASE CONFIGURATION USING SORTHEAP 2000
```

在某些情况下，要对一个内存配置参数启用自调整功能，还必须对另一个相关内存配置参数启用此功能。例如，仅当启用了 **locklist** 配置参数的自调整功能时，才允许启用 **maxlocks** 配置参数的自调整功能。同样，仅当启用了 **sortheap** 配置参数的自调整功能时，才能启用 **sheapthres_shr** 配置参数的自调整功能。这意味着，如果禁用 **locklist** 或 **sortheap** 参数的自调整功能，也将分别禁用 **maxlocks** 或 **sheapthres_shr** 参数的自调整功能。

通过将缓冲池设置为特定大小，可以禁用缓冲池的自调整功能。例如，下列语句将禁用 bufferpool1 的自调整功能:

```
ALTER BUFFERPOOL bufferpool1 SIZE 1000
```

## 确定启用了自调整功能的内存使用者

要查看配置参数控制的内存使用者的自调整设置，请使用下列其中一种方法。

- 要从命令行查看配置参数的自调整设置，请使用 GET DATABASE CONFIGURATION 命令并指定 SHOW DETAIL 参数。

  在输出中，可以启用自调整功能的内存使用者将分组到一起，如下所示:

```
 描述                                 参数                 当前值                延迟的值
 ------------------------------------------------------------------------------------------
 自调整内存                          (SELF_TUNING_MEM) =  ON (Active)           ON
 数据库共享内存大小（4KB）           (DATABASE_MEMORY) =  AUTOMATIC(37200)      AUTOMATIC(37200)
 最大锁定列表存储器（4KB）           (LOCKLIST) =         AUTOMATIC(7456)       AUTOMATIC(7456)
 每个应用程序的锁定列表百分比        (MAXLOCKS) =         AUTOMATIC(98)         AUTOMATIC(98)
程序包高速缓存大小（4KB）           (PCKCACHESZ) =       AUTOMATIC(5600)       AUTOMATIC(5600)
 共享排序的排序堆阈值（4KB）         (SHEAPTHRES_SHR) =   AUTOMATIC(5000)       AUTOMATIC(5000)
 排序列表堆（4KB）                   (SORTHEAP) =         AUTOMATIC(256)        AUTOMATIC(256)
```

- 也可以使用 db2CfgGet API 来确定是否启用调整功能。将返回下列值:

```
SQLF_OFF                    0
SQLF_ON_ACTIVE              2
SQLF_ON_INACTIVE            3
```

  SQLF_ON_ACTIVE 描述自调整功能已启用并处于活动状态的情况，而 SQLF_ON_INACTIVE 指示自调整功能已启用但当前处于不活动状态。

- 也可以在控制中心中的**数据库配置**窗口中查看配置设置。

要查看缓冲池的自调整设置，请使用下列其中一种方法。

- 要从命令行检索已启用自调整功能的缓冲池列表，请输入:

  ```
  db2 "SELECT BPNAME, NPAGES FROM SYSCAT.BUFFERPOOLS"
  ```

  当缓冲池启用了自调整功能时，该特定缓冲池的 SYSCAT.BUFFERPOOLS 目录视图中的 NPAGES 字段将设置为 -2。当自调整功能处于禁用状态时，NPAGES 字段将设置为缓冲池的当前大小。
- 要确定已启用自调整功能的缓冲池的当前大小，请按如下方式使用快照监视器并检查缓冲池的当前大小（bp_cur_buffsz 监视元素的值）:

  ```
  db2 GET SNAPSHOT FOR BUFFERPOOLS ON DB_NAME
  ```
- 要使用控制中心来查看缓冲池的自调整设置，请用鼠标右键单击缓冲池，并在对象详细信息窗格中查看缓冲池属性。

注意，内存调整器的反应受调整内存使用者的内存使用量所需时间的限制，这一点十分重要。例如，减小缓冲池大小的过程可能非常长，因此，为排序区域内存调整缓冲池内存大小所产生的性能优势可能不会立即体现。

## 分区数据库环境中的自调整内存

在分区数据库环境中使用自调整内存功能时，有一些因素决定该功能是否能适当地调整系统。

在分区数据库中启用自调整内存功能时，会将一个数据库分区指定为调整分区，所有内存调整决定都根据该数据库分区的内存和工作负载特征作出。一旦对调整分区作出调整决定，就会将内存调整分布到所有其他数据库分区上，以确保所有数据库分区都维持类似的配置。

单个调整分区模型要求只对具有类似内存需求的数据库分区使用该功能。可以使用下列准则确定您的分区数据库上是否要启用自调整内存功能。

### 建议在分区数据库中使用自调整的情况

当所有数据库分区都具有类似内存需求并且正在类似硬件上运行时，可以不进行任何修改就启用自调整内存功能。这些类型的环境共享下列特征:

- 所有数据库分区都在完全相同的硬件上，包括多逻辑节点平均分布在多个物理节点上
- 最佳或者接近最佳地分配数据
- 在数据库分区上运行的工作负载平均分布到各数据库分区上。这意味着没有一个数据库分区的一个或多个堆具有较多的内存需求。

在这种环境中，需要同等配置所有数据库分区，并且自调整内存将正确配置系统。

### 建议在分区数据库中小心使用自调整的情况

在环境中的大多数数据库分区具有类似内存需求并且正在类似硬件上运行的情况下，可以使用自调整内存功能，但在进行初始配置时要小心。这些系统可能有一组数据库分区用于数据，并且有一组少得多的协调程序分区和目录分区。在这些环境中，将协调程序分区和目录分区配置为与包含数据的数据库分区不同可能会有好处。

在此环境中，如果进行一些较小的设置，仍可以通过使用自调整内存功能获得好处。由于包含数据的数据库分区由大量数据库分区组成，因此应该在所有这些数据库分区上启用自调整，并应将这些数据库分区中的一个分区指定为调整分区。此外，由于目录分区和协调程序分区的配置可能不同，因此应在这些分区上禁用自调整内存功能。要在目录分区和协调程序分区上禁用自调整，将这些分区上的 *self_tuning_mem* 数据库配置参数更新为 OFF。

### 建议不要在分区数据库中使用自调整的情况

如果环境中的每个数据库分区的内存需求都不同，或者不同的数据库分区正在极不相同的硬件上运行，那么建议禁用自调整内存功能。这可以通过将所有分区上的 *self_tuning_mem* 数据库配置参数设置为 OFF 来实现。

### 比较不同数据库分区的内存需求

确定不同数据库分区的内存需求是否非常相似的最佳方法是查看快照监视器。如果所有分区上的下列快照元素相似（差别不超过 20%），那么认为这些分区相似。

通过发出 `get snapshot for database on <dbname>` 命令收集下列数据。

```
已分配的共享排序堆总数                  = 0
共享排序堆高水位标记                      = 0
后阈值排序数（共享内存）                  = 0
排序溢出数                              = 0

程序包高速缓存查询数                      = 13
程序包高速缓存插入数                      = 1
程序包高速缓存溢出数                      = 0
程序包高速缓存高水位标记（以字节计）      = 655360

散列连接数                              = 0
散列循环数                              = 0
散列连接溢出数                          = 0
小散列连接溢出数                        = 0
后阈值散列连接数（共享内存）            = 0

当前挂起的锁定数                        = 0
锁定等待数                              = 0
数据库等待锁定的时间（毫秒）            = 0
正在使用的锁定列表内存（以字节计）      = 4968
锁定升级数                              = 0
互斥锁定升级数                          = 0
```

通过发出 `get snapshot for bufferpools on <dbname>` 命令收集下列数据:

```
缓冲池数据逻辑读取数                = 0
缓冲池数据物理读取数                = 0
缓冲池索引逻辑读取数                = 0
缓冲池索引物理读取数                = 0
缓冲池总计读取时间（毫秒）          = 0
缓冲池总计写入时间（毫秒）          = 0
```

## 在分区数据库环境中使用自调整内存

在分区数据库环境中启用自调整功能之后，就会出现一个单独的数据库分区（称为*调整分区*），它监视内存配置的情况，并将任何配置更改传播到所有其他数据库分区以使所有参与数据库分区的配置保持一致。

调整分区是根据许多特征选择的，例如分区组中的数据库分区数以及已定义的缓冲池数。

- 要确定当前已指定为调整分区的数据库分区，请使用以下 ADMIN_CMD:

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

- 要更改调整分区，请使用以下 ADMIN_CMD:

  ```
  CALL SYSPROC.ADMIN_CMD( 'update stmm tuning dbpartitionnum <db_partition_num>')
  ```

  发出此命令时，将以异步方式或者在数据库下次启动时更新调整分区。

- 要让内存调整器自动重新选择调整分区，请对 <db_partition_num> 值输入 -1。

## 在 DPF 系统上启动内存调整器

由于自调整功能要求所有分区都处于活动状态才能正确地调整多分区系统上的内存，所以，在 DPF 环境中，仅当使用显式 ACTIVATE DATABASE 命令激活数据库时，才会启动内存调整器。

## 对给定数据库分区禁用自调整功能

- 要对一部分数据库分区禁用自调整功能，请将不想调整的数据库分区的 *self_tuning_mem* 配置参数设置为 OFF。
-

  要对特定数据库分区上由配置参数控制的一部分内存使用者禁用自调整功能，请在该数据库分区上将相关配置参数值或缓冲池大小设置为 MANUAL 或特定值。但是，建议使自调整配置参数值在所有正在运行的分区中一致。

- 要对某个数据库分区上的特定缓冲池禁用调整功能，请发出指定了大小值的 ALTER BUFFER POOL 命令并指定 PARTITIONNUM 参数值，以指示要禁用自调整功能的分区。

  使用 PARTITIONNUM 子句在特定数据库分区上指定大小的 ALTER BUFFERPOOL 语句将在 SYSCAT.SYSBUFFERPOOLNODES 目录中为给定缓冲池创建例外条目，如果例外条目已存在，那么此语句将更新该条目。如果缺省缓冲池大小设置为 AUTOMATIC，那么当此目录中的某个缓冲池存在例外条目时，该缓冲池将不会参与自调整操作。要除去例外条目，以便可以对缓冲池重新启用自调整功能:

  1. 通过发出 ALTER BUFFERPOOL 语句并将缓冲池大小设置为特定值来对此缓冲池禁用调整功能。
  2. 发出另一个 ALTER BUFFERPOOL 语句并指定 PARTITIONNUM 子句，以便将此数据库分区上缓冲池的大小设置为缺省缓冲池大小。
  3. 发出另一个 ALTER BUFFERPOOL 语句，将大小设置为 AUTOMATIC，从而启用自调整功能。

## 在不均匀的环境中启用自调整内存功能

理想情况下，数据应该均匀地分布在所有数据库分区中，每个分区上运行的工作负载的内存需求应该比较接近。如果数据分发不均匀，以致一个或多个数据库分区包含的数据显著多于或少于其他数据库分区，那么就不应该对这些不规则的数据库分区启用自调整功能。这也适用于不同数据库分区上的内存需求不均匀的情况。例如，如果只在一个分区上执行需要大量资源的排序操作，或者某些数据库分区使用的硬件与别的分区不同并且有更多的可用内存，就会发生这种情况。在此类环境中，仍然可以对某些数据库分区启用自调整功能。要在不均匀环境中利用自调整内存功能，请确定一组

具有相似数据和内存需求的数据库分区并对它们启用自调整功能。对于其余分区，应该手动进行内存配置。

## 配置内存和内存堆

借助简化的内存配置功能，可以通过使用大多数与内存相关的配置参数的缺省 AUTOMATIC 设置来配置 DB2 数据服务器需要的内存和内存堆，因此需要进行的调整工作更少。

简化的内存配置功能提供下列好处:

- 可以使用单个参数 **instance_memory** 来指定数据库管理器允许从其专用和共享内存堆中分配的所有内存。另外，可以使用 **appl_memory** 配置参数来控制 DB2 数据库代理程序分配的用于为应用程序请求提供服务的最大应用程序内存量。
- 不需要手动调整仅用于功能内存的参数。
- 可以使用“内存可视化器”来查询数据库管理器的专用内存堆和共享内存堆当前正在消耗的总内存。还可以使用 db2mtrk 命令来监视堆使用情况，并使用 ADMIN_GET_DBP_MEM_USAGE() 表函数来查询总内存消耗。
- 缺省 DB2 配置需要较少调整，将有利于您创建的新实例。

下表列示了其值缺省为 AUTOMATIC 设置的内存配置参数。需要时也可以动态配置这些参数。请注意，AUTOMATIC 设置的含义对于每个参数来说都不同，如最右边的列中所述。

*表 1. 其值缺省为 AUTOMATIC 的内存配置参数*

| 配置参数名称 | 描述 | AUTOMATIC 设置的含义 |
| --- | --- | --- |
| **appl_memory** | 控制 DB2 数据库代理程序分配的用于为应用程序请求提供服务的最大应用程序内存量。 | 如果强制施加了 **instance_memory** 限制，那么在数据库分区所分配的内存总量未超过 **instance_memory** 限制的情况下，AUTOMATIC 设置将允许所有应用程序内存请求。否则，只要有系统资源可用，它就会允许请求。 |
| **applheapsz** | 在版本 9.5 之前，此参数指的是为应用程序工作的每个数据库代理程序可以消耗的应用程序内存量。在版本 9.5 中，此参数指的是整个应用程序可以消耗的应用程序内存总量。对于 DPF、集中器或 SMP 配置，这意味着除非您使用 AUTOMATIC 设置，否则可能需要增大在先前发行版中使用的 **applheapsz** 值。 | AUTOMATIC 设置允许应用程序堆大小根据需要增大。如果存在 **appl_memory** 限制或 **instance_memory** 限制，就可能会强制施加限制。 |

*表 1. 其值缺省为 AUTOMATIC 的内存配置参数 (续)*

| 配置参数名称 | 描述 | AUTOMATIC 设置的含义 |
|---|---|---|
| **database_memory**（在版本 9.5 之前，缺省设置 AUTOMATIC 仅适用于 Windows 和 AIX 平台。对于版本 9.5，AUTOMATIC 是所有 DB2 服务器产品的缺省设置。） | 指定为数据库共享内存区域保留的共享内存量。 | 当内存调整器处于启用状态时，它确定数据库的整体内存需求并根据当前数据库需求增大或减小分配给数据库共享内存的内存量。 |
| **dbheap** | 确定数据库堆使用的最大内存。 | AUTOMATIC 设置允许数据库堆根据需要增大。如果存在 **database_memory** 限制或 **instance_memory** 限制，就可能会强制施加限制。 |
| **instance_memory** | 指定可以为数据库分区分配的最大内存量。 | AUTOMATIC 设置允许整个数据库管理器实例所消耗的内存总量根据需要增大，并且 STMM 确保有足够的系统内存可用，以防止过量使用内存。对于具有内存使用量限制的 DB2 数据库产品，AUTOMATIC 设置将根据计算值（75-95% 的 RAM）与许可证允许的内存使用量这两者当中的较小者来强制施加限制。有关何时将它强制施加为限制的详细信息，请参阅 **instance_memory**。 |
| **mon_heap_sz** | 确定分配给数据库系统监视器数据的内存量（以页计）。 | AUTOMATIC 设置允许监视器堆根据需要增大。如果存在 **instance_memory** 限制，就可能会强制施加限制。 |
| **stat_heap_sz** | 指示使用 RUNSTATS 命令收集统计信息时所使用的最大堆大小。 | AUTOMATIC 设置允许统计信息堆大小根据需要增大。如果存在 **appl_memory** 限制或 **instance_memory** 限制，就可能会强制施加限制。 |
| **stmtheap** | 指定语句堆的大小，SQL 或 XQuery 编译器将语句堆用作工作空间来编译 SQL 或 XQuery 语句。 | AUTOMATIC 设置允许语句堆根据需要增大。如果存在 **appl_memory** 限制或 **instance_memory** 限制，就可能会强制施加限制。 |

**注**: DBMCFG 和 DBCFG 管理视图检索所有数据库分区中当前连接的数据库的数据库管理器配置参数信息。对于 **mon_heap_sz**、**stmtheap** 和 **stat_heap_sz** 配置参数，此视图上的 DEFERRED_VALUE 列不会在数据库激活中持久存在。也就是说，在发出 get dbm cfg show detail 或 get db cfg show detail 命令时，查询输出将显示内存中已更新的值。

下表显示了在迁移或创建实例期间以及在迁移或创建数据库期间各个配置参数是否设置为缺省 AUTOMATIC 值。

*表 2. 在迁移和创建实例和数据库期间设置为 AUTOMATIC 的配置参数*

| 配置参数 | 在迁移或创建实例时设置为 AUTOMATIC | 在迁移数据库时设置为 AUTOMATIC | 在创建新数据库时设置为 AUTOMATIC |
|---|---|---|---|
| **applheapsz**[1] | | X | X |
| dbheap | | X | X |
| **instance_memory** | X | | |
| **mon_heap_sz**[1] | X | | |
| **stat_heap_sz**[1] | | X | X |
| **stmtheap**[1] | | | X |

在更改为简化的内存配置过程中，不推荐使用下列元素:

- 配置参数 **appgroup_mem_sz**、**groupheap_ratio** 和 **app_ctl_heap_sz**。这些配置参数已替换为新的 **appl_memory** 配置参数。
- db2mtrk 内存跟踪程序命令的 **-p** 参数。此选项列示专用代理程序内存堆，它已替换为列示所有应用程序内存消耗的 **-a** 参数。

“内存可视化器”使用新的 **appl_memory** 配置参数来显示数据库消耗的最大应用程序内存，并使用已更新的 **instance_memory** 配置参数来显示实例所消耗的最大内存。“内存可视化器”还显示允许 AUTOMATIC 设置的所有配置参数的值。对于版本 9.5 的数据库，“内存可视化器”中不会显示不推荐使用的配置参数的值，但是对于较早版本的数据库显示了这些配置参数的值。

分配快速通信管理器 (FCM) 共享内存时，将在该数据库分区的 **instance_memory** 限制中说明每个本地数据库分区在系统的 FCM 共享内存总大小中所占的份额。由于 FCM 内存的性质（未能分配 FCM 缓冲区将降低实例），所以 FCM 内存请求决不会因为 **instance_memory** 限制而失败。但是，如果不能从操作系统分配内存，那么它们可能会失败。如果 FCM 内存请求导致数据库分区超过其 **instance_memory** 限制，那么其他内存请求将失败，直到该分区的内存使用情况返回到小于 **instance_memory** 限制的级别为止。

## 代理程序和进程技术模型配置

版本 9.5 提供了一种较简单但更灵活的机制来配置与进程技术模型相关的参数。此简化的配置不需要定期调整这些参数，并减少了配置这些参数所需的时间和精力。它还无需通过关闭并重新启动 DB2 实例来使新值生效。

要允许动态和自动配置代理程序和内存，在激活实例时需要的内存资源会略有增多。

## 代理程序、进程模型和内存配置

DB2 数据服务器同时在 32 位和 64 位平台上使用多线程体系结构，这样为您提供了许多好处，例如，使用性增强、更好地共享资源、内存占用量减少以及所有操作系统上具有一致的线程技术体系结构。

## 配置跨多个分区的数据库

数据库管理器提供了多个分区中的所有数据库配置元素的单个视图。这意味着您可以更新或复位所有数据库分区中的数据库配置，而不必对每个数据库分区调用 db2_all 命令。

通过从数据库所在的任何分区只发出一个 SQL 语句或一个管理命令，即可更新多个分区中的该数据库配置。缺省情况下，用于更新或复位数据库配置的方法是*在所有数据库分区上*。

要实现命令脚本和应用程序的向后兼容性，您有下面三种选择:

- 使用 db2set 命令将 **DB2_UPDDBCFG_SINGLE_DBPARTITION** 注册表变量设置为 TRUE，如下所示:

  ```
  DB2_UPDDBCFG_SINGLE_DBPARTITION=TRUE
  ```

  **注**: 设置该注册表变量不适用于使用 ADMIN_CMD 过程发出的 UPDATE DATABASE CONFIGURATION 或 RESET DATABASE CONFIGURATION 请求。

- 对 UPDATE DATABASE CONFIGURATION 或 RESET DATABASE CONFIGURATION 命令或者 ADMIN_CMD 过程使用 **DBPARTITIONNUM** 参数。例如，要更新所有数据库分区上的数据库配置，请按如下所示调用 ADMIN_CMD 过程:

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

  要更新单个数据库分区，请按如下所示调用 ADMIN_CMD 过程:

  ```
  CALL SYSPROC.ADMIN_CMD
    ('UPDATE DB CFG DBPARTITIONNUM 10 USING sortheap 1000')
  ```

- 对 db2CfgSet API 使用 **DBPARTITIONNUM** 参数。**db2Cfg** 结构中的标志指示数据库配置的值是否将应用于单个数据库分区。如果设置一个标志，那么还必须提供 **DBPARTITIONNUM** 值，例如:

  ```
  #define db2CfgSingleDbpartition          256
  ```

  如果未设置 db2CfgSingleDbpartition 值，那么该数据库配置的值将应用于所有数据库分区，除非对用于设置数据库管理器或数据库配置参数的 db2CfgSet API 将 **DB2_UPDDBCFG_SINGLE_DBPARTITION** 注册表变量设置为 TRUE，或者将 *versionNumber* 设置为低于版本 9.5 的版本号的任意版本号。

将数据库迁移到版本 9.5 时，现有数据库配置参数在迁移后通常会保留它们的值。但是，已添加了使用缺省值的新参数，并将一些现有参数设置为新的版本 9.5 缺省值。有关对现有数据库配置参数的更改的详细信息，请参阅《*迁移指南*》中的“DB2 服务器行为更改”主题。对于已迁移数据库发出的任何后续更新或复位数据库配置请求，将使用版本 9.5 中用于更新或复位配置请求的方法进行处理。

对于现有更新或复位命令脚本，前面提到的规则同样适用: 可以使用版本 9.5 之前的方法，也可以修改脚本以包括 UPDATE DATABASE CONFIGURATION 或 RESET DATABASE CONFIGURATION 命令的 **DBPARTITIONNUM** 选项，也可以设置 **DB2_UPDDBCFG_SINGLE_DBPARTITION** 注册表变量。

对于调用 db2CfgSet API 的现有应用程序，必须使用版本 9.5 方法。如果要使用版本 9.5 之前的版本的方法，那么可以设置 **DB2_UPDDBCFG_SINGLE_DBPARTITION** 注

册表变量，或者修改应用程序以使用版本 9.5 版本号调用该 API，包括新的 db2CfgSingleDbpartition 标志以及用于更新或复位特定数据库分区的数据库配置的新 **dbpartitionnum** 字段。

**注:** 如果您发现数据库配置值不一致，那么可以单独地更新或复位每个数据库分区。

### 共享文件句柄表

线程数据库管理器为每个数据库和在该数据库上工作的所有代理程序维护单个共享文件句柄表，以便对同一文件发出的 I/O 请求不需要重新打开和关闭文件。

在版本 9.5 之前，文件句柄表由每个 DB2 代理程序单独维护，并且每个代理程序文件句柄表的大小由 **maxfilop** 配置参数控制。从版本 9.5 起，数据库管理器对整个数据库维护单个共享文件句柄表，以便在相同数据库文件上工作的所有代理程序之间可以共享同一文件句柄。因此，**maxfilop** 配置参数用来控制共享文件句柄表的大小。

正是由于此更改，**maxfilop** 配置参数具有新的缺省值以及新的最小值和最大值。在数据库迁移期间，**maxfilop** 配置参数将自动设置为新的缺省值。

### 在受防护方式进程中运行供应商库函数

数据库管理器支持用于执行诸如数据压缩、TSM 备份和日志数据归档等任务的受防护方式进程中的供应商库函数。

在版本 9.5 之前，供应商库函数、供应商实用程序或例程在代理进程中运行。从版本 9.5 开始，因为 DB2 数据库管理器本身是多线程应用程序，所以不再是线程安全的供应商库函数可能会导致内存或堆栈被毁坏，或者更严重时可能会导致 DB2 数据库中的数据被毁坏。由于这些原因，每次调用供应商实用程序时就会创建一个新的受防护方式进程，并且供应商库函数或例程在此受防护方式进程内运行。这不会导致性能急剧下降。

**注:** 受防护方式功能不可用于 Windows 平台。

## 自动存储器

自动存储器简化了表空间的存储管理。创建数据库时，可以指定数据库管理器将用来存放表空间数据的存储路径。然后，当您创建并填充表空间时，数据库管理器将管理这些表空间的容器和空间分配。

### 自动存储器表空间

当在未启用自动存储器的数据库中创建表空间时，必须指定 MANAGED BY SYSTEM 或 MANAGED BY DATABASE 子句. 使用这些子句会导致分别创建系统管理的空间（SMS）表空间或数据库管理的空间（DMS）表空间。在两种情况下都必须提供显式的容器列表。

如果数据库启用了自动存储器，那么存在其他选项: 您可以指定 MANAGED BY AUTOMATIC STORAGE 子句，或者省略 MANAGED BY 子句（这意味着将使用自动存储器）. 在这种情况下，您不需要提供容器定义，因为数据库管理器会自动指定容器。

以下是一些创建自动存储器表空间的示例语句:

```
CREATE TABLESPACE TS1
CREATE TABLESPACE TS2 MANAGED BY AUTOMATIC STORAGE
CREATE TEMPORARY TABLESPACE TEMPTS
CREATE USER TEMPORARY TABLESPACE USRTMP MANAGED BY AUTOMATIC STORAGE
CREATE LONG TABLESPACE LONGTS
```

虽然自动存储器表空间类型好像是另外的表空间类型，但它实际上只是现有 SMS 和 DMS 类型的扩展。如果您要创建一个常规表空间或大型表空间，那么会将它创建为具有文件容器的 DMS 表空间。如果您要创建一个用户或系统临时表空间，那么会将它创建为具有目录容器的 SMS 表空间。

**注**: 此行为在数据库管理器的将来版本中可能会有所不同。

与这些容器相关联的名称具有以下格式:

*storage path*/*instance name*/NODE####/*database name*/T#######/C#######.*EXT*

其中:

*storage path*
是与数据库相关联的存储路径

*instance name*
是创建了数据库的实例

*database name*
是数据库的名称

**NODE####**
是数据库分区号（例如，NODE0000）

**T#######**
是表空间标识（例如，T0000003）

**C#######**
是容器标识（例如，C0000012）

*EXT* 是基于要存储的数据类型的扩展名:

**CAT** 系统目录表空间

**TMP** 系统临时表空间

**UTM** 用户临时表空间

**USR** 用户或常规表空间

**LRG** 大型表空间

## 常规和大型自动存储器表空间与 DMS 表空间之间的区别

常规和大型自动存储器表空间是作为 DMS 表空间创建的，与 DMS 表空间相关联的所有规则和行为都适用。但是，存储器的管理方式存在一些差异，如下表中所示:

*表 3. 管理非自动存储器和自动存储器表空间时存在的差异*

| 非自动存储器 | 自动存储器 |
| --- | --- |
| 创建表空间时必须显式提供容器列表。 | 创建表空间时不能提供容器列表；但是，数据库管理器将自动指定和分配容器。 |

*表 3. 管理非自动存储器和自动存储器表空间时存在的差异 (续)*

| 非自动存储器 | 自动存储器 |
| --- | --- |
| 缺省情况下关闭自动重新调整表空间的大小（AUTORESIZE 设置为 NO）。 | 缺省情况下打开自动重新调整表空间的大小（AUTORESIZE 设置为 YES）。 |
| 不能使用 INITIALSIZE 子句来指定表空间的初始大小。 | 可以使用 INITIALSIZE 子句来指定表空间的初始大小。 |
| 可以使用 ALTER TABLESPACE 语句（指定 ADD、DROP 和 BEGIN NEW STRIPE SET 等等）来执行容器操作。 | 不能执行容器操作，因为数据库管理器将管理空间。 |
| 可以使用重定向复原操作来重新定义与表空间相关联的容器。 | 不能使用重定向复原操作来重新定义与表空间相关联的容器，因为数据库管理器将管理空间。 |

正如前面表中所提到的那样，当您创建常规或大型自动存储器表空间时，可以指定初始大小作为 CREATE TABLESPACE 语句的一部分，如以下示例中所示:

```
CREATE TABLESPACE TS1 INITIALSIZE 100 M
```

如果未指定初始大小，那么数据库管理器使用缺省值（即，32MB）。

要创建具有给定大小的表空间，数据库管理器将在存储路径中创建文件容器。如果各路径之间空间分布不均匀，那么创建容器时它们可能具有不同的大小。因此，所有存储路径上的可用空间大致相等非常重要。

如果对表空间启用自动调整大小，那么在使用表空间中的空间时，数据库管理器会自动扩展现有容器并添加新的容器（使用分割集）。无论是扩展还是添加容器，都不会进行重新平衡。

## 自动重新调整表空间的大小

在自动存储器表空间启用自动重新调整大小的情况下，数据库管理器会通过添加容器的新分割集来自动处理文件系统变满的情况。

数据库系统中可以存在两种类型的表空间: 系统管理的空间（SMS）和数据库管理的空间（DMS）。与 SMS 表空间相关联的容器是文件系统目录，而这些目录中的文件会随着表空间中对象的增多而增大。。文件会逐渐增大，直到达到其中一个容器的文件系统限制或者达到数据库的表空间大小限制（请参阅 SQL 和 XML 限制）。

DMS 表空间由文件容器或原始设备容器组成，它们的大小是在将容器指定给表空间时设置的。当容器中的所有空间都已被使用时，那么认为表空间将满。但是，与 SMS 表空间不同，您可以使用 ALTER TABLESPACE 语句来添加或扩展容器，从而允许将更多的存储空间提供给表空间。DMS 表空间还有一项称为*自动调整大小*的功能: 当可以自动调整大小的 DMS 表空间中的空间被消耗时，数据库系统可能会对该表空间扩展一个或多个文件容器。SMS 表空间具有类似于自动增长的功能，但术语“自动调整大小”专门用于 DMS。

自动重新调整表空间的大小具有下列含义:

- 启用了自动调整大小的表空间具有版本 8.2.1 或更早发行版不能识别的相关元数据。对这些版本尝试使用启用了自动调整大小的表空间的数据库会产生故障（很有可能会返回 SQL0980C 或 SQL0902C 错误）。如果您尝试连接至数据库或者尝试复原数

据库，那么可能会发送错误。如果启用了表空间自动重新调整大小，那么对这些表空间禁用自动调整大小功能就会除去元数据，从而允许对版本 8.2.1 或更早发行版使用该数据库。

- 如果禁用自动调整大小功能，后来又启用此功能，那么与 INCREASESIZE 和 MAXSIZE 相关联的值会丢失。
- 不能对使用原始设备容器的表空间使用此功能，也不能将原始设备容器添加至可以自动调整大小的表空间。尝试执行这些操作会产生错误（SQL0109N）。如果需要添加原始设备容器，那么必须首先禁用此功能。
- 重定向复原操作不能更改容器定义以包括原始设备容器。尝试执行这种操作会产生错误（SQL0109N）。
- 由于最大大小限制了数据库管理器自动增大表空间的方式，所以最大大小也限制了您可增大表空间的方式。也就是说，当执行向表空间添加空间的操作时，生成的大小必须小于或等于最大大小。可以使用 ALTER TABLESPACE 语句的 ADD、EXTEND、RESIZE 或 BEGIN NEW STRIPE SET 子句来添加空间。

### 启用和禁用自动调整大小功能

缺省情况下，不会对 DMS 表空间启用自动调整大小功能。以下语句将创建不启用自动调整大小功能的 DMS 表空间:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
    USING (FILE '/db2files/DMS1' 10 M)
```

要启用自动调整大小功能，对 CREATE TABLESPACE 语句指定 AUTORESIZE YES 子句:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
    USING (FILE '/db2files/DMS1' 10 M) AUTORESIZE YES
```

在创建 DMS 表空间之后，还可以使用带有 AUTORESIZE 子句的 ALTER TABLESPACE 语句来启用或禁用自动调整大小功能:

```
ALTER TABLESPACE DMS1 AUTORESIZE YES
ALTER TABLESPACE DMS1 AUTORESIZE NO
```

有另外两个属性 MAXSIZE 和 INCREASESIZE 与自动调整大小的表空间相关联:

#### 最大大小（MAXSIZE）

CREATE TABLESPACE 语句的 MAXSIZE 子句定义表空间的最大大小。例如，以下语句创建可增长至 100 兆字节（如果数据库有多个数据库分区，那么是每个数据库分区的大小）的表空间:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
    USING (FILE '/db2files/DMS1' 10 M)
    AUTORESIZE YES MAXSIZE 100 M
```

MAXSIZE NONE 子句指定表空间没有最大限制。表空间会逐渐增大，直到达到文件系统限制或表空间限制（请参阅 SQL 和 XML 限制）。如果您不指定 MAXSIZE 子句，那么在启用了自动调整大小功能的情况下没有最大值限制。

使用 ALTER TABLESPACE 语句来更改已经启用了自动调整大小功能的表空间的 MAXSIZE 值，如下列示例中所示:

```
ALTER TABLESPACE DMS1 MAXSIZE 1 G
ALTER TABLESPACE DMS1 MAXSIZE NONE
```

如果指定了最大大小，那么数据库管理器强制使用的实际值可能会比指定的值略小，这是因为数据库管理器会尝试使容器的增长保持一致。

### 增大大小（INCREASESIZE）

当表空间中没有可用扩展数据块但是请求了一个或多个扩展数据块时，CREATE TABLESPACE 语句的 INCREASESIZE 子句将定义用来增大表空间的空间量。可以将值指定为显式大小或者指定为百分比，如下列示例中所示:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
    USING (FILE '/db2files/DMS1' 10 M)
    AUTORESIZE YES INCREASESIZE 5 M

CREATE TABLESPACE DMS1 MANAGED BY DATABASE
    USING (FILE '/db2files/DMS1' 10 M)
    AUTORESIZE YES INCREASESIZE 50 PERCENT
```

百分比值意味着每当需要增大表空间时都要计算增大大小（由 INCREASESIZE 值指定），并且增大量基于增大时表空间大小所占的百分比。例如，如果表空间大小是 20 MB 而 INCREASESIZE 值是 50%，那么表空间第一次将增大 10 MB（增大到 30 MB），下一次将增大 15 MB。

如果在启用自动调整大小功能时未指定 INCREASESIZE 子句，那么数据库管理器将确定要使用的适当值，该值在表空间的生存期内可能会发生变化。与 AUTORESIZE 和 MAXSIZE 一样，可以使用 ALTER TABLESPACE 语句更改 INCREASESIZE 的值。

如果指定了增大大小，那么数据库管理器将使用的实际值可能会与您提供的值稍微有所不同。对所用值进行这种调整是为了使表空间中各容器的增大保持一致。

## 如何扩展表空间

对于可以自动调整大小的表空间，当所有现有空间都已被使用并请求了更多空间时，数据库管理器会尝试增大该表空间的大小。数据库管理器确定可以扩展表空间中的哪些容器以便不需要进行重新平衡。数据库管理器只扩展位于表空间图（该图描述表空间的存储器布局）的最后范围内的那些容器，并且对它们扩展相同的数量。

例如，考虑如下语句:

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE
    USING (FILE 'C:\TS1CONT' 1000, FILE 'D:\TS1CONT' 1000,
                  FILE 'E:\TS1CONT' 2000, FILE 'F:\TS1CONT' 2000)
    EXTENTSIZE 4
    AUTORESIZE YES
```

请记住，数据库管理器将每个容器的一小部分（一个扩展数据块）用于元数据，以下是根据 CREATE TABLESPACE 语句为表空间创建的表空间图。（表空间图是表空间快照输出的一部分。）

表空间图:

```
范围    分割    分割区     最大         最大     起始      结束     调节    容器
编号     集      偏移   扩展数据块      页     分割区    分割区
[   0] [   0]   0        995        3983      0       248       0     4 (0,1,2,3)
[   1] [   0]   0        1495       5983     249      498       0     2 (2,3)
```

表空间图表明标识为 2 或 3 的容器（E:\TS1CONT 和 F:\TS1CONT）是仅有的在图最后面范围内的容器。因此，当数据库管理器自动扩展此表空间中的容器时，它只扩展这两个容器。

**注**: 如果创建表空间时让所有容器的大小都相同，那么图中只有一个范围。在这种情况下，数据库管理器扩展每一个容器。要防止限制为只扩展一小部分容器，创建表空间时使各容器大小相等。

如先前所讨论的那样，可以指定对表空间最大大小的限制，也可以指定值 NONE 以便不限制表空间的增大。如果指定了 NONE 或者无限制，那么上限由文件系统限制或表空间限制定义；数据库管理器不会尝试让表空间大小增大到超过上限。但是，在达到上限之前，尝试增大容器可能会因为文件系统已满而失败。在这种情况下，数据库管理器不会再增大表空间大小，但是会向应用程序返回“空间不足”情况。解决此情况有两种方法:

- 增大已满文件系统上可用的空间量。
- 对表空间执行一些容器操作，以使这些容器不再位于表空间图的最后。完成此任务的最简易方法是将新的分割集添加至具有一组新容器的表空间，最佳实践是确保所有容器的大小相同。可以使用带有 BEGIN NEW STRIPE SET 子句的 ALTER TABLESPACE 语句来添加新的分割集。通过添加新的分割集，就会将新的范围添加至表空间图。借助新的范围，数据库管理器自动尝试扩展的容器就会处于此新的分割集中，而旧的容器保持不变。

  **注**: 当暂挂用户启动的容器操作或者正在执行后续生新平衡时，会禁用自动调整大小功能，直到落实了操作或重新平衡完成为止。

例如，对于 DMS 表空间，假定一个表空间具有三个大小相同的容器，并且每个容器都位于它自己的文件系统上。当对表空间执行一些操作时，数据库管理器就会自动扩展这三个容器。最后，其中一个文件系统变满了，对应的容器就不能再增大了。如果该文件系统上不能再提供更多可用空间，那么必须对表空间执行容器操作，使得存在问题的容器不再处于表空间图的最后范围内。在这种情况下，您可以添加新的分割集并指定两个两个容器（仍然具有空间的每个文件系统上一个），也可以指定更多或更少容器（再次确保要添加的每个窗口大小一样并且要使用的每个文件系统上有足够的空间）。当数据库管理器尝试增大表空间的大小时，它现在将尝试扩展此新分割集中的容器而不会尝试扩展旧容器。

### 监视

有关对 DMS 表空间自动调整大小的信息是作为表空间监视器快照输出的一部分显示的。增大大小和最大大小值也包括在输出中，如以下样本中所示:

```
 启用自动调整大小                = Yes 或 No
当前表空间大小（字节）           = ###
最大表空间大小（字节）           = ### 或 NONE
增加大小（字节）                 = ###
增加大小（百分比）               = ###
上一次成功调整大小的时间         = YYYY/MM/DD HH:MM:SS.SSSSSS
 上一次调整大小尝试失败          = Yes 或 No
```

## 自动存储器数据库

缺省情况下，数据库管理器会将所有数据库都作为“自动存储器”数据库来创建。要创建不是“自动存储器”数据库的数据库，请在发出 CREATE DATABASE 命令时指定 AUTOMATIC STORAGE NO。

启用自动存储器数据库有一套一个或多个存储路径与之关联。可以将表空间定义为*由自动存储器管理*，而它的容器由数据库管理器根据那些存储路径来指定和分配。

仅当创建数据库时才能对该数据库启用自动存储器。同样，不能对最初设计为使用自动存储器的数据库禁用自动存储器。

缺省情况下，将所有数据库作为自动存储器数据库来创建。要创建不是自动存储器数据库的数据库，请在发出 CREATE DATABASE 命令时指定 **AUTOMATIC STORAGE NO**。

以下是禁用自动存储器的一些示例:

```
CREATE DATABASE ASNODB1 AUTOMATIC STORAGE NO
CREATE DATABASE ASNODB2 AUTOMATIC STORAGE NO ON X:
```

显式或隐式启用自动存储器的一些示例:

```
CREATE DATABASE DB1
CREATE DATABASE DB2 AUTOMATIC STORAGE YES ON X:
CREATE DATABASE DB3 ON /data/path1, /data/path2
CREATE DATABASE DB4 ON D:\StoragePath DBPATH ON C:
```

根据所用的语法，数据库管理器抽取下列有关存储位置的两部分信息:

- 数据库路径（数据库管理器将数据库的各种控制文件存储在其中）:
  - 如果您指定 **DBPATH ON**，那么这指示数据库路径。
  - 如果不指定 **DBPATH ON**，那么 **ON** 中列示的第一个路径指示数据库路径（和存储路径）。
  - 如果既未指定 **DBPATH ON** 又未指定 **ON**，那么使用 **dftdbpath** 数据库管理器配置参数来确定数据库路径。
- 存储路径（数据库管理器在其中创建自动存储器表空间容器）:
  - 如果指定了 **ON**，那么列示的所有路径都是存储路径。
  - 如果未指定 **ON**，那么将存在单个存储路径，它被设置为 **dftdbpath** 数据库管理器配置参数的值。

对于前面显示的示例，下表概述了所使用的数据库路径和存储路径:

*表 4. 自动存储器数据库路径和存储路径*

| **CREATE DATABASE 命令** | **数据库路径** | **存储路径** |
|---|---|---|
| CREATE DATABASE DB1 AUTOMATIC STORAGE YES | **dftdbpath** 配置参数的值 | **dftdbpath** 配置参数的值 |
| CREATE DATABASE DB2 AUTOMATIC STORAGE YES ON X: | X: | X: |
| CREATE DATABASE DB3 ON /data/path1, /data/path2 | /data/path1 | /data/path1, /data/path2 |
| CREATE DATABASE DB4 ON D:\StoragePath DBPATH ON C: | C: | D:\StoragePath |

提供的存储路径必须存在并且可以访问。在分区数据库环境中，将在每个数据库分区上使用相同的存储路径。除非将数据库分区表达式用作存储路径名的一部分，否则不能为特定数据库分区指定一组唯一的存储路径。将数据库分区表达式用作存储路径名的一部分允许在存储路径中反映数据库分区号，这样生成的路径名在每个数据库分区上都不相同。

使用自变量 $N（即，在 $N 前面添加一个空格）来指示数据库分区表达式。可以在存储路径中的任何位置使用数据库分区表达式，并且可以指定多个数据库分区表达式。使用空格字符来表示数据库分区表达式的结束区表达式以空格字符结束；在对数据库

分区表达式求值以后，该空格后跟的所有内容都将追加至存储路径。如果存储路径中的数据库分区表达式后面没有空格字符，那么假定其余字符串是该表达式的一部分。下表列示了 $N 自变量唯一有效的格式。从左向右按运算符求值，% 表示模数运算符。示例中的数据库分区号为 10。

*表 5. 数据库分区表达式*

| 语法 | 示例 | 值 |
|---|---|---|
| [blank]$N | " $N" | 10 |
| [blank]$N+[number] | " $N+100" | 110 |
| [blank]$N%[number] | " $N%5" | 0 |
| [blank]$N+[number]%[number] | " $N+1%5" | 1 |
| [blank]$N%[number]+[number] | " $N%4+2" | 4 |

以下是使用数据库分区表达式的示例:

```
CREATE DATABASE TESTDB ON "/path1ForNode $N",
  "/path2ForNode $N" DBPATH ON "/dbpathForNodes"
```

以下是嵌入在路径中的数据库分区表达式的示例:

```
CREATE DATABASE TESTDB ON "/path1ForNode $N",
  "/path2ForNode $N suffix" DBPATH ON "/dbpathForNodes"
```

**注**: 无论数据库分区表达式是在 **DBPATH ON** 中显式指定的，还是通过在第一个存储路径中使用数据库分区表达式隐式指定的，数据库分区表达式在数据库路径中都无效。

计算出给定数据库分区的存储路径的可用空间之后，数据库管理器将检查该存储路径中是否存在下列目录或安装点，并使用找到的第一个目录或安装点:

```
storage path/instance name/NODE####/database name
storage path/instance name/NODE####
   storage path/instance name
storage path
```

其中:

*storage path*
是与数据库相关联的存储路径

*instance name*
是数据库所在的实例

**NODE####**
是数据库分区号（例如，NODE0000 或 NODE0001）

*database name*
是数据库的名称

可以将文件系统安装在存储路径中的某个位置，并且数据库管理器将意识到可用于表空间容器的实际可用空间大小可能和与存储路径目录本身相关联的那个大小不同。

请考虑以下示例: 在一台物理计算机上存在两个逻辑数据库分区，并且有一个存储路径: /db2data。每个数据库分区都可以使用此存储路径，但是您可能想通过为每个数据库分区创建单独的文件系统来将每个数据库分区中的数据分隔开。文件系统安装在

/db2data/*instance*/NODE#### 中。在存储路径中创建容器并确定可用空间时，数据库管理器不会检索 /db2data 的可用空间信息，但是会检索相应 /db2data/*instance*/NODE#### 目录的可用空间信息。

每当创建数据库时都会创建三个缺省表空间。如果您不提供显式表空间定义作为 CREATE DATABASE 命令的一部分，那么会将表空间创建为自动存储器表空间。

创建数据库之后，可以使用 ALTER DATABASE 语句的 ADD STORAGE 子句来对该数据库添加新的存储路径，如以下示例中所示:

```
ALTER DATABASE ADD STORAGE ON '/data/path3', '/data/path4'
```

### 将自动存储路径添加至启用了自动存储器的数据库

通过使用 ALTER DATABASE 语句，可以将自动存储路径添加至启用了自动存储器的数据库。仅当创建数据库时才能对该数据库启用自动存储器。

对多分区数据库环境添加存储路径时，每个数据库分区上都必须存在存储路径。如果指定的路径在每个数据库分区上都不存在，那么将回滚该语句。

要将存储路径添加至现有数据库，请发出以下 ALTER DATABASE 语句:

```
ALTER DATABASE PATH pathname
```

## 自动存储器限制

在决定是否创建使用自动存储器的数据库时，应考虑一些限制。

- 创建数据库之后，不能对它禁用或启用自动存储器。
- 存储路径必须是绝对路径名。它们可以是 Windows 操作系统上的路径或盘符。数据库路径必须是盘符。 最大路径长度是 175 个字符。
- 对于分区数据库，必须在每个数据库分区上使用同一组存储路径（除非您使用数据库分区表达式）。
- 无论数据库分区表达式是使用 CREATE DATABASE 命令的 **DBPATH ON** 选项显式指定的，还是通过在第一个存储路径中使用数据库分区表达式隐式指定的，数据库分区表达式在数据库路径中都无效。

# 创建压缩字典

可以自动构建或手动构建为行压缩而启用的表的压缩字典。

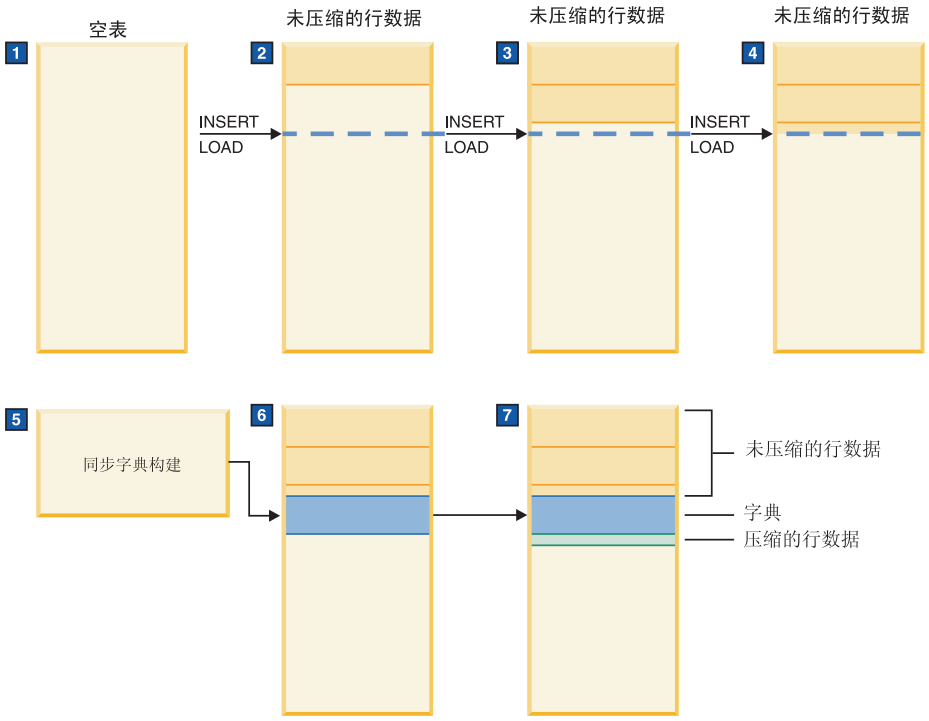
### 自动创建字典

从 DB2 版本 9.5 开始，如果满足下列每个条件，那么会自动创建压缩字典:

- 表的 COMPRESS 属性设置为 YES。当您创建表时，可以使用 CREATE TABLE 语句的 **COMPRESS YES** 选项将此属性设置为 YES; 还可以在 ALTER TABLE 语句中使用该选项将现有表改变为使用压缩功能。
- 该表尚不存在压缩字典
- 表的大小达到一定程度，有足够的数据用于构造重复数据的字典。

如果仍然启用了压缩，那么就会使用该压缩字典对后来移入该表的数据进行压缩。

下图显示了自动创建压缩字典的过程:




1. 由于表是空的而未创建压缩字典。
2. 数据是使用插入或装入操作插入表中的，并且保持未压缩。
3. 随着更多数据被插入或装入表中，它保持未压缩。
4. 如果 COMPRESS 属性设置为 YES，那么在达到阈值之后就会自动触发字典的创建过程。
5. 创建了字典。
6. 已将字典追加至表。
7. 从此时起，数据就是压缩的。

**要点**：仅对随后插入该表中以创建字典的行进行压缩。对于创建该字典之前已存在的行，除非更改了这些行，或者手动重建了该字典，否则它们将保持未压缩状态。

**限制**：未压缩 XML 列中的数据。

## 手动创建字典

尽管支持压缩的表增大到足够大时会自动创建字典，但是如果不存在压缩字典，您也可以强制创建压缩字典，或者使用带有 **RESETDICTIONARY** 选项的 REORG TABLE 命令来复位现有压缩字典。如果该表中至少有一行数据，此命令就会强制创建压缩字典。对表进行重组是一项脱机操作；使用“自动创建字典”功能的一项好处是构建字典时该表仍然处于联机状态。

您可以使用带有 **ROWCOMPESTIMATE** 选项的 INSPECT 命令，而不使用 REORG TABLE 命令来强制创建新的字典。如果该表尚无压缩字典，那么此命令将创建一个新的

压缩字典。此方法的优势在于该表仍然处于联机状态。但是，会对您随后添加的行进行压缩，在运行 INSPECT 命令之前已存在的行将保持未压缩状态，直到您对表执行重组为止。

### 复位压缩字典

无论是自动还是手动创建压缩字典，字典均为静态；在构建之后也不会更改。当您添加或更新行时，将根据当前压缩字典中存在的数据来压缩这些行。此行为在许多情况下都存在。例如，以用于维护城市自来水公司客户帐户的数据库中的表为例。这样一个表中可能有 STREET_ADDRESS、CITY、PROVINCE、TELEPHONE_NUM、POSTAL_CODE 和 ACCOUNT_TYPE 等列。如果使用这种表中的数据构建了压缩字典，即使它只是一个适当大小的表，也可能存在足够多的重复信息，因此对它进行行压缩将节省大量空间。这是因为不同客户之间可能有许多共同数据，例如 CITY、POSTAL_CODE 和 PROVINCE 都相同，而 STREET_ADDRSS 或 TELEPHONE_NUM 列的某些部分相同。

但是，其他表可能会随时间推移而显著更改。以一个用于存储零售数据的范围分区表为例，如下所示:

- 每个分区将存储一年中特定月份的数据，
- 在月末，使用 ALTER TABLE 语句的 ATTACH PARTITION 子句将具有所给定月份的销售数据的分区转入该表中。

这种情况下，在某个月份（比如说，四月份）创建的压缩字典可能不会反映后续月份重复的销售数据。在表中数据随时间推移而显著更改的情况下，您可能希望使用带有 **RESETDICTIONARY** 选项的 REORG TABLE 命令来复位压缩字典。

## 行压缩

行压缩（有时称为*深度压缩*）通过将各行之间重复的值模式替换为更短的符号字符串来压缩数据行。

使用行压缩的主要好处在于，可以在更少的空间中存储数据，从而显著降低存储成本。此外，因为您耗用存储器的速度更慢，所以将来花钱购买更多存储器的时间也会相应推迟。

除了降低成本以外，通过压缩还可以提高性能。通过执行较少的 I/O 操作就可以对已压缩的数据执行许多查询，这是因为每次从磁盘中读取时将读入更多数据。同样，可以在缓冲池中高速缓存更多数据，因此将提高缓冲池的命中率。（但是，行压缩也存在缺点，即，对数据进行压缩和解压缩需要耗用额外的 CPU 周期。）数据行压缩节省的存储量和对性能的影响与数据库中数据的特征、数据库的布局和调整以及应用程序工作负载相关。查询优化器的成本模型中包括解压缩成本。

最后，因为行压缩可以减小数据库大小，所以执行备份和复原操作将使用更少的空间，并且运行速度也更快。

本主题的其余部分将讨论下列内容:


- 第 45 页的『压缩的工作方式』
- 第 45 页的『对哪些数据进行压缩？』
- 第 45 页的『启用或禁用行压缩』
- 第 45 页的『UPDATE 活动对日志和已压缩表的影响』.

### 压缩的工作方式

行压缩使用基于静态字典的压缩算法来逐行压缩数据。字典用来将表行中重复的字节模式映射至更小的符号；这时，这些符号替换表行中更长的字节模式。压缩字典与表数据行一起存储在表的数据对象部分。

### 对哪些数据进行压缩?

可对存储在基本表行和日志记录中的数据进行行压缩。

**限制**: 未压缩 XML 列中的数据。

### 启用或禁用行压缩

要使用行压缩，您必须具有 DB2 存储优化功能部件的许可证。通过将表的 COMPRESS 属性设置为 YES 来压缩表数据。可以在创建表时通过在 CREATE TABLE 语句中包括 **COMPRESS YES** 选项来设置此属性；也可以在 ALTER TABLE 语句中使用此选项将现有表改变为使用压缩功能。启用压缩功能之后，用于对表添加数据的操作（例如，INSERT、LOAD INSERT 或 IMPORT INSERT 操作）可以使用行压缩功能。

**要点**: 对表启用行压缩时，就对整个表启用了行压缩，即使此表由多个表分区组成也是如此。

要对表禁用压缩功能，请使用带有 **COMPRESS NO** 选项的 ALTER TABLE 语句；将不会对您以后添加的行进行压缩。要对整个表进行解压缩，必须使用 REORG TABLE 命令来执行表重组。

### UPDATE 活动对日志和已压缩表的影响

根据 UPDATE 活动以及数据行中更新更改的位置，可能会增加日志的使用。有关更新日志记录和列顺序的信息，请参阅『第 221 页的『对列进行排序以使更新日志记录最少』』。

如果行大小增大，那么该行的新版本可能不适合放在当前数据页上。而且，行的新映像将存储在溢出页上。要将创建指针溢出记录的次数降到最低限度，应增大每一页中在使用 ALTER TABLE 语句的 **PCTFREE** 选项进行重组之后保留的可用空间所占的百分比。例如，如果在启用压缩功能之前 PCTFREE 设置为 5%，那么可以在启用压缩功能时将它更改为 10%。对于频繁更新的数据来说，此建议尤其重要。

## 自动收集统计信息

DB2 优化器使用目录统计信息来确定任何给定查询的最佳访问方案。如果有关表或索引的统计信息已过时或者不完整，那么可能会导致优化器选择不是最佳的方案，并且会降低执行查询的速度。但是，决定要为给定的工作负载收集哪些统计信息是很复杂的事情，并且使这些统计信息保持最新是一项很花费时间的任务。

通过使用自动收集统计信息功能（它是 DB2 的自动表维护功能的一部分），可以让 DB2 数据库管理器确定是否需要更新数据库统计信息。可以在编译语句时使用实时统计信息（RTS）功能来执行自动收集统计信息操作，也可以在后台运行 RUNSTATS 实用程序来收集统计信息。可以在实时收集统计信息功能被禁用时启用后台收集统计信

息。必须启用后台收集统计信息才能启用实时收集统计信息。当您创建新的数据库时，缺省情况下就会启用后台自动收集统计信息。实时自动收集统计信息是由动态配置参数 **auto_stmt_stats** 启用的。

## 了解异步收集和实时收集统计信息

通过运行 RUNSTATS 实用程序，可以采用同步或异步方式自动收集统计信息。*异步*收集在后台执行。当实时统计信息功能启用后，还可以在编译语句时*同步*收集统计信息。启用实时收集统计信息功能之后，还可以使用由索引和数据管理器维护的元数据来生成统计信息。*生成*意味着派生或创建统计信息，而不是作为正常 RUNSTATS 活动的一部分来收集统计信息。例如，如果知道一个表中的行数、页大小和平均行宽，就可以知道该表中包含的行数。在某些情况下，并未真正派生统计信息，而是由索引和数据管理器在维护统计信息，并且可以直接将这些统计信息存储在目录中。例如，索引管理器将在每个索引中维护叶子页数和级别数。

查询优化器根据查询需求和表更新活动的数量来确定统计信息的收集方式。根据执行的更新、插入或删除操作的次数来衡量表更新活动。

实时收集统计信息由优化 SQL 语句之前该语句的需求确定。这样可以更及时地收集更准确的统计信息。准确的统计信息可以产生更好的查询执行方案并提高性能。未启用实时收集统计信息功能时，将每隔两小时执行一次异步收集统计信息操作。要为某些应用程序提供准确的统计信息，此收集频率可能不够高。

在启用实时收集统计信息功能之后，仍将每隔两个小时执行一次异步收集统计信息检查操作。在下列情况下，实时收集统计信息功能还将引起发出异步收集请求:

- 表活动频率对于同步收集而言不够高，但对于异步收集而言却又过高
- 由于表太大，因此同步收集统计信息操作进行了采样
- 已生成同步统计信息
- 同步收集统计信息操作由于超出收集时间而失败

最多可以同时处理两个异步请求，但必须针对不同的表处理这些请求。一个请求由实时统计信息发起，另一个请求由异步收集统计信息检查发起。

通过使用下面几种方法，可以将自动收集统计信息功能对性能的影响降到最低:

- 通过使用已调速的 RUNSTATS 实用程序执行异步收集统计信息。调速功能根据当前数据库活动来控制 RUNSTATS 实用程序消耗的资源数量: 随着数据库活动的增加，RUNSTATS 实用程序的运行速度将变慢，从而减少了它的资源需求。
- 对于每个查询，同步收集统计信息操作的时间限制为 5 秒。此值可由 RTS 优化准则控制。如果同步收集操作超出时间限制，那么将提交异步收集请求。
- 同步收集统计信息操作不会将统计信息存储在系统目录中。而是，此操作将统计信息存储在统计信息高速缓存中，以后再通过异步操作存储在系统目录中。这样可避免更新系统目录所产生的开销和可能的锁定争用。后续 SQL 编译请求可以使用统计信息高速缓存中的统计信息。
- 对于每个表，只能执行一个同步收集统计信息操作。其他需要执行同步收集统计信息操作的代理程序将在可能的情况下生成统计信息，并继续编译语句。在分区数据库环境中也会强制执行此行为，此环境中的不同数据库分区中的代理程序可能需要同步统计信息。

- 缺省情况下，收集的关于同步和异步操作的统计信息是基本表统计信息和分发信息，以及使用采样的详细索引统计信息。（发出 RUNSTATS 命令时指定了 WITH DISTRIBUTION 和 SAMPLED DETAILED INDEXES ALL 选项。）可以通过启用统计信息概要分析来定制收集的统计信息类型，统计信息概要分析使用有关先前的数据库活动的信息来确定数据库工作负载需要的统计信息。还可以通过为特定表创建您自己的统计信息概要文件来定制收集的关于该表的统计信息类型。
- 只对缺少统计信息或具有高级别活动（根据执行的更新、插入或删除操作的次数来衡量）的表进行统计信息收集。即使表符合统计信息收集条件，也不会收集同步统计信息，除非查询优化需要这些信息。在某些情况下，查询优化器可以不使用统计信息来选择访问方案。
- 对于异步收集统计信息检查，还对大型表（超过 4000 页）进行采样以确定高级表活动是否更改了统计信息。只有更改了统计信息，才会收集这种大型表的统计信息。
- 对于异步收集统计信息，自动将 RUNSTATS 实用程序安排在维护策略定义中指定的联机维护时间段内运行。此策略还指定自动收集统计信息作用域内的一组表，这进一步减少了不必要的资源消耗。
- 同步收集和生成统计信息操作不在维护策略定义中所指定的联机维护时间段之后发生，这是因为同步请求必须立即执行并且收集时间有限。同步收集和生成统计信息操作遵循以下策略: 指定自动收集统计信息作用域内的一组表。
- 当正在执行自动收集统计信息时，受影响的表仍然可用于常规数据库活动（更新、插入或删除操作）。
- 对于异步收集统计信息，通过使用基于目录的收集方法来运行 SYSPROC.NNSTAT 存储过程，从而自动刷新昵称统计信息。不收集昵称的实时统计信息（同步或生成的）。

对常规表、具体化查询表 (MQT) 和已声明的全局临时表（DGTT）执行实时同步统计信息收集。不收集 DGTT 的异步统计信息。这表示实时统计信息处理将不发起 DGTT 的异步请求。

不会对下列对象执行自动收集统计信息（同步或异步）操作:

- 统计视图
- 标记为 VOLATILE 的表（在 SYSCAT.TABLES 目录视图中设置了 VOLATILE 字段的表）
- 通过直接对 SYSSTAT 目录视图发出 UPDATE 语句手动更新了其统计信息的表

  当手动修改表的统计信息时，数据库管理器假定您负责维护其统计信息。对于已手动更新其统计信息的表，为了允许数据库管理器重新考虑和维护该表的统计信息，请使用 RUNSTATS 命令收集统计信息或指定在使用 LOAD 命令时收集统计信息。在版本 9.5 之前创建且在迁移之前已手动更新其统计信息的表不受影响，而且其统计信息将由数据库管理器自动维护，一直到您手动更新其统计信息为止。

不会对下列对象生成统计信息:

- 统计视图
- 通过直接对 SYSSTAT 目录视图发出 UPDATE 语句手动更新了其统计信息的表。注意，如果未启用实时收集统计信息功能，仍将对已手动更新其统计信息的表生成某些统计信息。

在分区数据库环境中，将在单个数据库分区上收集统计信息然后进行推测。数据库管理器始终在数据库分区组的第一个数据库分区上收集统计信息（同步和异步）。

至少在数据库激活 5 分钟之后才会执行非实时统计信息收集活动。

将对静态和动态 SQL 执行实时统计信息处理。

已使用 IMPORT 命令截断的表将自动重组为具有旧统计信息。

对于已收集其统计信息的表，同步和异步自动收集统计信息操作将使引用了这些表并已进行高速缓存的动态语句失效。这样做是为了能够使用最新统计信息来重新优化已进行高速缓存的动态语句。

数据库停用时，异步收集统计信息操作可能会中断。如果未使用 ACTIVATE DATABASE 命令或 API 显式激活数据库，那么在最后一个用户与数据库断开连接时，将停用数据库。如果操作中断，那么错误消息可能会记录在 DB2 诊断日志文件中。要避免异步自动统计信息收集操作中断，请显式激活数据库。

## 实时统计信息和说明处理

不会对刚刚使用说明工具说明（不是执行）的查询进行实时处理。下表概述了 CURRENT EXPLAIN MODE 专用寄存器的不同值的行为。

*表 6. 作为 CURRENT EXPLAIN MODE 专用寄存器值的应变量的实时收集统计信息操作*

| CURRENT EXPLAIN MODE 值 | 是否考虑实时收集统计信息 |
|---|---|
| YES | 是 |
| EXPLAIN | 否 |
| NO | 是 |
| REOPT | 是 |
| RECOMMEND INDEXES | 否 |
| EVALUATE INDEXES | 否 |

## 自动收集统计信息和统计信息高速缓存

DB2 版本 9.5 引入了统计信息高速缓存，以便使以同步方式收集的统计信息可用于所有查询。此高速缓存是目录高速缓存的一部分。在分区数据库环境中，此高速缓存仅存在于目录数据库分区中。目录高速缓存可以存储同一个 SYSTABLES 对象的多个条目，这将增大所有数据库分区中的目录高速缓存大小。在启用实时收集统计信息功能的情况下，应考虑增大 **catalogcache_sz** 数据库配置参数的值。

从 DB2 版本 9 开始，可以使用配置顾问程序来确定新数据库的初始配置。配置顾问程序建议将 **auto_stmt_stats** 数据库配置参数设置为 ON。

## 自动收集统计信息和统计概要文件

根据表的有效统计概要文件同步和异步收集统计信息，但下列情况例外:

- 为了使同步收集统计信息操作的开销降至最低，数据库管理器可以通过进行采样来收集统计信息。在这种情况下，采样率和方法可能与统计概要文件中指定的采样率和方法不同。

- 同步收集统计信息可以选择生成统计信息，但是可能无法生成在统计概要文件中指定的所有统计信息。例如，无法生成 COLCARD、HIGH2KEY 和 LOW2KEY 之类的列统计信息，除非该列在某些索引中是前导列。

如果同步收集统计信息操作不能收集统计概要文件中指定的所有统计信息，那么将提交异步收集请求。

尽管实时收集统计信息功能旨在使收集统计信息的开销降至最低，但是请先在测试环境中尝试使用此功能，以确保不会对性能产生负面影响。对某些联机事务处理（OLTP）方案使用此功能可能会对性能产生负面影响，对查询运行时间设置了上限的情况下尤其如此。

## 启用自动收集统计信息

要进行高效率的数据访问和获得最佳工作负载性能，具有准确而完整的数据库统计信息是很关键的。使用自动进行表维护功能的自动收集统计信息功能来更新和维护相关的数据库统计信息。可以选择通过使用以下方法在串行环境（也就是单个数据库分区在单个处理器上运行的环境）中增强此功能：收集查询数据并生成统计信息概要文件，这可以帮助 DB2 自动收集工作负载需要的一组准确的统计信息。在 MPP 环境、某些联合环境或者启用了分区内并行性的环境中，此选项不可用。

要启用自动收集统计信息:

1. 使用“配置自动维护”向导或者命令行来配置数据库实例:
   - 要使用“配置自动维护”向导:
     a. 通过以下两种方法来打开此向导：一种方法是在“控制中心”中右键单击数据库对象；另一种方法是在“运行状况中心”中右键单击数据库实例。
     b. 从弹出窗口中选择**配置自动维护**。在此向导中，可以启用自动收集统计信息、指定想要自动从其中收集统计信息的表以及指定用于执行 RUNSTATS 实用程序的维护窗口。
   - 要使用命令行，请将下列每个配置参数都设置为 ON:
     – **AUTO_MAINT**
     – **AUTO_TBL_MAINT**
     – **AUTO_RUNSTATS**
2. 可选：要启用自动生成统计信息概要文件，应将下面两个配置参数都设置为 ON:
   - **AUTO_STATS_PROF**
   - **AUTO_PROF_UPD**
3. 可选：要启用收集实时统计信息，将 **AUTO_STMT_STATS** 配置参数设置为 ON。如果此配置参数设置为 ON，那么每当需要表统计信息来优化查询时，就会在编译语句时自动编译这些统计信息。

## 配置顾问程序

可以使用“配置顾问程序”来获取有关缓冲池大小、数据库配置参数和数据库管理器配置参数的初始值的建议。

要使用“配置顾问程序”，对现有数据库指定 AUTOCONFIGURE 命令，或者将 AUTOCONFIGURE 指定为 CREATE DATABASE 命令的一个选项。要配置数据库，您必须具有 SYSADM、SYSCTRL 或 SYSMAINT 权限。

可以显示建议值，或者使用 CREATE DATABASE 命令的 APPLY 选项来应用建议值。建议是根据您提供的输入和顾问程序收集的系统信息生成的。

“配置顾问程序”建议的值只是针对每个实例具有一个数据库的情况。如果想要在多个数据库上使用此顾问程序，那么每个数据库必须属于一个单独的实例。

## 使用“配置顾问程序”调整配置参数

“配置顾问程序”通过建议修改某些配置参数并为它们提供建议值来帮助调整性能和平衡每个实例中单个数据库的内存需求。创建数据库时会自动运行“配置顾问程序”。

要禁用此功能或显式启用它，可在创建数据库之前使用 db2set 命令，如下所示:

```
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
```

要定义多个配置参数的值和确定应用程序的这些参数的作用域，使用 AUTOCONFIGURE 命令并指定下列其中一个选项:

- NONE，表示未应用任何值
- DB ONLY，表示仅应用数据库配置和缓冲池值
- DB AND DBM，表示应用所有参数以及它们的值

**注:** 即使您在运行 CREATE DATABASE 命令时自动启用“配置顾问程序”，仍然可以指定 AUTOCONFIGURE 命令选项。如果您在运行 CREATE DATABASE 命令时未启用“配置顾问程序”，那么稍后可以手动运行“配置顾问程序”。

## 生成数据库配置建议

创建数据库时会自动运行“配置顾问程序”。还可以通过在命令行处理器（CLP）中指定 AUTOCONFIGURE 命令或者通过调用 db2AutoConfig API 来运行“配置顾问程序”。

要使用 CLP 来请求配置建议，请输入以下命令:

```
AUTOCONFIGURE
     USING input_keyword param_value
     APPLY value
```

以下是 AUTOCONFIGURE 命令的一个示例，它请求根据有关如何使用数据库的输入提供配置建议，但指定不应该应用这些建议:

```
DB2 AUTOCONFIGURE USING
     MEM_PERCENT 60
     WORKLOAD_TYPE MIXED
     NUM_STMTS 500
     ADMIN_PRIORITY BOTH
     IS_POPULATED YES
     NUM_LOCAL_APPS 0
     NUM_REMOTE_APPS 20
     ISOLATION RR
     BP_RESIZEABLE YES
APPLY NONE
```

## 示例: 使用“配置顾问程序”请求配置建议

此方案说明从命令行运行“配置顾问程序”以生成建议并显示“配置顾问程序”生成的结果。

要运行“配置顾问程序”:

1. 通过从命令行指定以下命令来连接至 PERSONL 数据库:

   ```
   DB2 CONNECT TO PERSONL
   ```

2. 从 CLP 发出 AUTOCONFIGURE 命令并指定该数据库的使用方式。如以下示例中所示，将 **APPLY** 选项的值设置为 NONE，以表明您想查看配置建议但是不应用这些建议:

   ```
   DB2 AUTOCONFIGURE USING
        MEM_PERCENT 60
        WORKLOAD_TYPE MIXED
        NUM_STMTS 500
        ADMIN_PRIORITY BOTH
        IS_POPULATED YES
        NUM_LOCAL_APPS 0
        NUM_REMOTE_APPS 20
        ISOLATION RR
        BP_RESIZEABLE YES
   APPLY NONE
   ```

   如果您不太确定此命令的参数的值，那么可以忽略此参数值，这种情况下将使用缺省值。最多可以传递 10 个参数但不附带值: MEM_PERCENT 和 WORKLOAD_TYPE 等等，如前面示例中所示。

由 AUTOCONFIGURE 命令生成的建议按表的形式显示在屏幕上:

*表 7. 配置顾问程序样本输出: 第一部分*

数据库管理器配置的先前值和应用的值

| 描述 | 参数 | 当前值 | 建议值 |
|---|---|---|---|
| 应用程序支持层堆大小（4KB） | (ASLHEAPSZ) = | 15 | 15 |
| 内部通信缓冲区数（4KB） | (FCM_NUM_BUFFERS) = | AUTOMATIC | AUTOMATIC |
| 启用分区内并行性 | (INTRA_PARALLEL) = | NO | NO |
| 最大查询并行度 | (MAX_QUERYDEGREE) = | ANY | 1 |
| 代理程序池大小 | (NUM_POOLAGENTS) = | 100（已计算） | 200 |
| 池中的初始代理程序数 | (NUM_INITAGENTS) = | 0 | 0 |
| 最大请求者 I/O 块大小（以字节计） | (RQRIOBLK) = | 32767 | 32767 |
| 排序堆阈值（4KB） | (SHEAPTHRES) = | 0 | 0 |

*表 8. 配置顾问程序样本输出（续）*

```
        数据库管理器配置的先前值和应用的值
描述                                  参数                 当前值            建议值
-------------------------------------------------------------------------------------
缺省应用程序堆（4KB）                (APPLHEAPSZ) =          256              256
目录高速缓存大小（4KB）              (CATALOGCACHE_SZ) =    (MAXAPPLS*4)      260
更改的页数阈值                       (CHNGPGS_THRESH) =        60              80
数据库堆（4KB）                      (DBHEAP) =              1200             2791
并行度                               (DFT_DEGREE) =            1               1
缺省表空间扩展数据块大小（页数）     (DFT_EXTENT_SZ) =        32              32
缺省预取大小（页数）                 (DFT_PREFETCH_SZ) =      AUTOMATIC       AUTOMATIC
缺省查询优化类                     (DFT_QUERYOPT) =          5               5
锁定列表的最大存储器（4KB）          (LOCKLIST) =             100             AUTOMATIC
日志缓冲区大小（4KB）                (LOGBUFSZ) =              8               99
日志文件大小（4KB）                  (LOGFILSIZ) =            1000            1024
主日志文件数                         (LOGPRIMARY) =            3               8
辅助日志文件数                       (LOGSECOND) =             2               3
最大活动应用程序数                   (MAXAPPLS) =             AUTOMATIC       AUTOMATIC
每个应用程序的锁定百分比列表         (MAXLOCKS) =              10             AUTOMATIC
组落实计数                           (MINCOMMIT) =              1               1
异步页清除程序的数目                 (NUM_IOCLEANERS) =         1               1
I/O 服务器数                         (NUM_IOSERVERS) =          3               4
程序包高速缓存大小（4KB）            (PCKCACHESZ) =           (MAXAPPLS*8)     1533
软检查点前回收的日志文件的百分比     (SOFTMAX) =                100             320
排序列表堆（4KB）                    (SORTHEAP) =               256            AUTOMATIC
语句堆（4KB）                        (STMTHEAP) = 4096          4096
统计信息堆大小（4KB）                (STAT_HEAP_SZ) =           4384            4384
实用程序堆大小（4KB）                (UTIL_HEAP_SZ) =           5000            113661
自调整内存                           (SELF_TUNING_MEM) =         ON              ON
自动 runstats                        (AUTO_RUNSTATS) =           ON              ON
共享排序的排序堆阈值（4KB）          (SHEAPTHRES_SHR) =         5000            AUTOMATIC
```

*表 9. 配置顾问程序样本输出（续）*

```
        缓冲池的先前值和应用的值
描述                                  参数                 当前值            建议值
-------------------------------------------------------------------------------------
IBMDEFAULTBP                         缓冲池大小 =           -2               340985

DB210203I AUTOCONFIGURE 成功完成。可能更改了
数据库管理器或数据库配置值。必须重新启动实例
所有更改才能生效。在新的配置参数生效之后，您还可能要
新的配置参数生效，以便使用新值。
```

如果您同意所有建议值，那么重新发出 AUTOCONFIGURE 命令，但是指定您想使用 APPLY 选项来应用建议值，或者使用 UPDATE DATABASE MANAGER CONFIGURATION 和 UPDATE DATABASE CONFIGURATION 命令来更新各个配置参数。

## 实用程序调速

实用程序调速调整各种维护实用程序对性能的影响，以便在生产期间可以同时运行这些维护实用程序。虽然缺省情况下定义了影响策略（此设置允许实用程序采用已调速方式运行），但是如果您想对实用程序调速，那么在运行实用程序时必须设置影响优先级（每个清除程序都通过此设置来指示它的调速优先级）。

调速系统确保已调速实用程序尽可能频繁地运行而不违反影响策略。可以调速统计信息收集、备份操作、重新平衡操作和异步索引清除。

通过设置 **util_impact_lim** 配置参数来定义影响策略。

清除程序与实用程序调速功能集成。缺省情况下，每个索引清除程序的实用程序影响优先级为 50（可接受的值为 1 到 100，0 表示无调速功能）。可以使用 SET UTIL_IMPACT_PRIORITY 命令或 db2UtilityControl API 来更改此优先级。

## 异步索引清除

异步索引清除（AIC）是在使索引条目失效的操作之后的延迟索引清除。根据索引的类型，条目可以是行标识（RID）或块标识（BID）。无论是哪种标识，这些条目都将由索引清除程序除去，索引清除程序在后台异步运行。

AIC 加快从分区表拆离数据分区的速度。如果分区表包含一个或多个非分区索引，那么将启动 AIC。在这种情况下，AIC 将除去引用了已拆离的数据分区和任何已伪删除的条目的所有非分区索引条目。在清除所有索引之后，将从系统目录中除去与已拆离的数据分区相关联的标识。

**注:** 如果分区表定义了从属具体化查询表 (MQT)，那么 AIC 要在执行了 SET INTEGRITY 操作之后才启动。

当 AIC 正在进行时，将维护正常表访问。访问索引的查询将忽略尚未清除的任何无效条目。

在大多数情况下，对与分区表关联的每个非分区索引启动一个清除程序。内部任务分发守护程序负责将 AIC 任务分发给适当的数据库分区并指定数据库代理程序。

分发守护程序和清理代理程序都是内部系统应用程序。它们出现在 LIST APPLICATION 输出中，应用程序名称分别为 **db2taskd** 和 **db2aic**。为了防止意外中断，不能强制执行系统应用程序。只要数据库是活动的，分发守护程序就一直处于联机状态。清除程序在完成清理之前保持活动状态。如果在进行清理时取消激活了数据库，那么当您重新激活数据库时将继续进行 AIC。

### 性能

AIC 对性能产生很小影响。

需要进行瞬时行锁定测试以确定是否落实了伪删除的条目。但是，由于从未获取锁定，因此并行性不会受影响。

每个清除程序都获取最小表空间锁定（IX）和表锁定（IS）。如果清除程序确定其他应用程序正在等待这些锁定，就会释放这些锁定。如果发生这种情况，那么清除程序就会暂挂处理 5 分钟。

清除程序与实用程序调速功能集成。缺省情况下，每个清除程序的实用程序影响优先级为 50。可以使用 SET UTIL_IMPACT_PRIORITY 命令或 db2UtilityControl API 来更改此优先级。

### 监视

可以使用 LIST UTILITIES 命令来监视 AIC。每个索引清除程序都作为一个单独的实用程序出现在监视器中。

以下示例使用命令行处理器（CLP）界面演示了当前数据库分区中 WSDB 数据库中的 AIC 活动:

```
$ db2 list utilities show detail

标识                                 = 2
类型                                 = ASYNCHRONOUS INDEX CLEANUP
数据库名称                           = WSDB
分区号                               = 0
描述                                 = 表: USER1.SALES, 索引: USER1.I2
开始时间                             = 12/15/2005 11:15:01.967939
状态                                 = 正在执行
调用类型                             = 自动
正在调速:
      优先级                           = 50
进度监控:
            总计工作                     = 5 页
            已完成的工作                 = 0 页
            开始时间                     = 12/15/2005 11:15:01.979033

标识                                 = 1
类型                                 = ASYNCHRONOUS INDEX CLEANUP
数据库名称                           = WSDB
分区号                               = 0
描述                                 = 表: USER1.SALES, 索引: USER1.I1
开始时间                             = 12/15/2005 11:15:01.978554
状态                                 = 正在执行
调用类型                             = 自动
正在调速:
      优先级                           = 50
进度监控:
            总计工作                     = 5 页
            已完成的工作                 = 0 页
            开始时间                     = 12/15/2005 11:15:01.980524
```

在本示例中，有两个清除程序正在对 USERS1.SALES 表进行操作。一个清除程序正在处理索引 I1；另一个清除程序是处理索引 I2。进度监视部分显示需要清除的估计总索引页数和当前的干净索引页数。

状态字段指示清除程序的当前状态。通常，状态为“正在执行”。如果清除程序正在等待被指定给可用的数据库代理程序，或者如果清除程序由于锁定争用而临时暂挂，那么清除程序可能处于“正在等待”状态。

**注**: 因为每个数据库分区将标识仅指定给该数据库分区上的任务，所以不同数据库分区上的不同任务可能具有相同的实用程序标识。

## MDC 表的异步索引清除

可以使用异步索引清除（AIC）来提高转出删除的性能。转出删除是一种从多维集群（MDC）表中删除符合条件的数据块的有效方法。AIC 是指在执行使索引条目失效的操作之后延迟清除索引。

在执行标准转出删除期间，将在执行删除操作时同步清除索引。对于包含许多记录标识（RID）索引的表，很大一部分删除时间花在除去索引键，索引键引用了被删除的表行。可以通过指定在落实删除之后就清除这些索引来提高转出速度。

要对 MDC 表利用 AIC，需要显式启用*延迟索引清除转出*机制。可以采用两种方法来指定延迟转出：将注册表变量 **DB2_MDC_ROLLOUT** 设置为 DEFER 以及发出 SET CURRENT MDC ROLLOUT MODE 语句。在执行延迟索引清除转出期间，各个块被标记为已转出，但是未更新 RID 索引，直到落实事务之后才会进行更新。在删除期间仍然会清除块标识（BID）索引，这是因为它们并不需要执行行级别处理。

在落实转出删除时将调用转出 AIC；如果数据库已关闭，那么在重新启动该数据库之后首次访问表时也会调用转出 AIC。在进行 AIC 时，对索引进行的所有查询（包括将访问被清除的索引的那些查询）都会执行。

每个 MDC 表都有一个协调清除程序。对于多个转出的索引清除都合并在清除程序中。清除程序将对每个 RID 索引产生一个清除代理程序，各个清除代理程序将并行更新 RID 索引。清除程序还与实用程序调速功能集成。缺省情况下，每个清除程序的实用程序影响优先级为 50（可接受的值为 1 到 100，0 表示无调速功能）。可以使用 SET UTIL_IMPACT_PRIORITY 命令或 db2UtilityControl API 来更改此优先级。

## 监视

因为在完成清除之后才能复用 MDC 表上已转出的块，所以监视延迟索引清除转出的进度将很有用。使用 LIST UTILITIES 监视器命令来显示被清除的每个索引的实用程序监视器条目。还可以通过延迟索引清除转出（BLOCKS_PENDING_CLEANUP）并使用 SYSPROC.ADMIN_GET_TAB_INFO_V95 表函数来查询当前正在清除的表中的块数。要查询数据库级别的 MDC 表块暂挂清除数，请使用 GET SNAPSHOT 命令。

在 LIST UTILITIES 命令的以下样本输出中，进度由每个索引中已被清除的页数指示。输出中列示的每个阶段都表示正在对表清除的一个 RID 索引。

```
db2 LIST UTILITIES SHOW DETAILS 输出。
标识                                  = 2
类型                                  = MDC ROLLOUT INDEX CLEANUP
数据库名称                            = WSDB
分区号                                = 0
描述                                  = TABLE.<schema_name>.<table_name>
开始时间                              = 06/12/2006 08:56:33.390158
状态                                  = 正在执行
调用类型                              = 自动
正在调速:
   优先级                             = 50
进度监控:
  估计已完成的百分比          = 83
   阶段号                             = 1
      描述                            = <schema_name>.<index_name>
      工作总计                        = 13 页
      已完成的工作                    = 13 页
      开始时间                        = 06/12/2006 08:56:33.391566
   阶段号                             = 2
      描述                            = <schema_name>.<index_name>
      工作总计                        = 13 页
      已完成的工作                    = 13 页
      开始时间                        = 06/12/2006 08:56:33.391577
   阶段号                             = 3
      描述                            = <schema_name>.<index_name>
```

```
总计工作                    = 9 页
已完成的工作                = 3 页
开始时间                    = 06/12/2006 08:56:33.391587
```

# 第 4 章 实例

*实例*是逻辑数据库管理器环境，您可以在此环境中对数据库进行编目和设置配置参数。根据需要，可以在同一台物理服务器上创建多个实例，该服务器为每个实例提供唯一的数据库服务器环境。

**注**: 在 Linux® 和 UNIX 操作系统上以非 root 用户身份安装时，在安装 DB2 产品期间将创建单个实例。不能创建更多实例。

可使用多个实例来执行下列操作:

- 将一个实例用作开发环境，将另一个实例用作生产环境。
- 调整一个实例以用作特定的环境。
- 限制对敏感信息的访问。
- 控制每个实例中对 SYSADM、SYSCTRL 和 SYSMAINT 权限的指定。
- 优化每个实例的数据库管理器配置。
- 限制实例失败所带来的影响。如果一个实例失败，那么只影响一个实例。其他实例可继续正常运行。

对于多个实例来说:

- 每个实例都需要额外的系统资源（虚拟内存和磁盘空间）。
- 由于要管理其他的实例，因此增加了管理工作量。

实例目录存储着与一个数据库实例相关的所有信息。实例目录一旦创建，就不能更改其位置。该目录包含:

- 数据库管理器配置文件
- 系统数据库目录
- 节点目录
- 节点配置文件（`db2nodes.cfg`）
- 包含调试信息（例如，异常或寄存器转储或用于 DB2 数据库进程的调用堆栈）的任何其他文件。

**术语**:

**位宽** 用于地址虚拟内存的位数: 最常见的是 32 位和 64 位。此术语可用于指实例、应用程序代码或外部例程代码的位宽。32 位应用程序的含义与 32 位宽应用程序的含义相同。

**32 位 DB2 实例**
包含所有 32 位二进制的 DB2 实例，包括 32 位共享库和可执行文件。

**64 位 DB2 实例**
包含 64 位共享库和可执行文件、所有 32 位客户机应用程序库（包括客户机和服务器）以及 32 位外部例程支持（仅包括服务器实例）的 DB2 实例。

© Copyright IBM Corp. 1993, 2010

## 设计实例

DB2 数据库是在数据库服务器上的 DB2 实例内创建的。在同一物理服务器上创建多个实例将为每个实例提供唯一的数据库服务器环境。

例如，可以在同一台机器上维护测试环境和生产环境，也可以为每个应用程序创建一个实例，然后专门对实例将为其提供服务的应用程序微调每个实例，或者为了保护敏感数据，可以将工资单数据库存储在它自己的实例中，以便同一服务器上其他实例的所有者看不到工资单数据。

安装过程将创建缺省 DB2 实例，该实例由 DB2INSTANCE 环境变量定义。这是用于大多数操作的实例。但是，可以在安装后创建（或删除）实例。

为环境确定并设计实例时，应注意每个实例控制对一个或多个数据库的访问。实例内的每个数据库都被指定了唯一的名称、具有它自己的一组系统目录表（这些表用于跟踪在数据库内创建的对象），并且具有它自己的配置文件。每个数据库还有它自己的一组可授予的权限和特权，它们控制用户与存储在该数据库中的数据和数据库对象的交互方式。图 2 显示了系统、实例和数据库之间的分层关系。

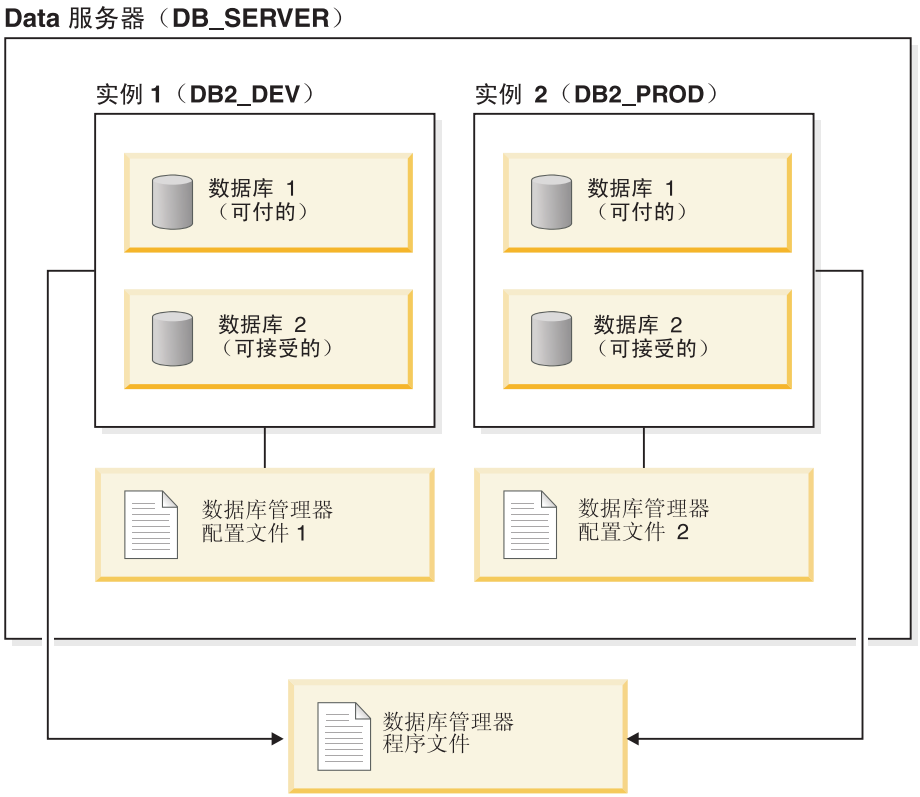



*图 2. DB2 系统、实例和数据库之间的分层关系*

还需要了解另一种特定类型的实例，它称为 *DB2 管理服务器*（DAS）。DAS 是一个特殊的 DB2 管理控制点，它用于仅帮助其他 DB2 服务器上的管理任务。如果要使用“客户机配置助手”来发现远程数据库或随 DB2 产品一起提供的图形工具（例如，控制中心或任务中心），那么 DAS 必须正在运行。即使有多个实例，DB2 数据库服务器中也只有一个 DAS。

创建实例后，可以与任何其他可用的实例（包括其他系统上的实例）连接。连接后，可以执行只能在实例级执行的维护和实用程序任务，例如，创建数据库、强制断开应用程序与数据库的连接、监视数据库活动，或者更改与该特定实例关联的数据库管理器配置文件的内容。

## 缺省实例

在 DB2 安装过程中，如果没有其他实例的名称为“DB2”，那么将创建一个名为“DB2”的数据库管理器初始实例。如果安装了 DB2 V8，并且已升级到版本 9.1 或版本 9.5，那么缺省实例为 『DB2_01』。

在 Linux 和 UNIX 上，只要符合命名规则，就可以随意命名初始实例。实例名用于设置目录结构。

要想立即使用此实例，在安装期间设置下列各项:

- 环境变量 **DB2INSTANCE** 设置为“DB2”。
- 注册表变量 **DB2INSTDEF** 设置为“DB2”。

这些设置将建立“DB2”作为缺省实例。可以更改在缺省情况下使用的实例，但首先必须创建一个附加实例。

在使用数据库管理器之前，必须更新每个用户的数据库环境，以便该环境可以访问实例并运行 DB2 数据库程序。这适用于所有用户（包括管理用户）。

在 Linux 和 UNIX 操作系统上，提供了样本脚本文件来帮助您设置数据库环境。这些文件有: 用于 Bourne 或 Korn shell 程序的 `db2profile` 以及用于 C shell 的 `db2cshrc`。这些脚本位于实例所有者主目录下的 `sqllib` 子目录中。实例所有者或属于该实例的 SYSADM 组的任何用户可为该实例的所有用户定制脚本。使用 `sqllib/userprofile` 和 `sqllib/usercshrc` 来为每个用户定制脚本。

在实例创建期间，将创建空文件 `sqllib/userprofile` 和 `sqllib/usercshrc` 以允许您添加自己的实例环境设置。安装 DB2 修订包时，在更新实例期间将覆盖 `db2profile` 和 `db2cshrc` 文件。如果不想使用 `db2profile` 或 `db2cshrc` 脚本中的新环境设置，那么可以使用相应的用户脚本覆盖它们，该脚本是在 `db2profile` 或 `db2cshrc` 脚本末尾调用的。在实例迁移（使用 db2imigr 命令）期间，将覆盖 user 脚本以使您所作的环境修改仍然有效。

样本脚本包含用于执行以下操作的语句:

- 将以下目录添加到现有的搜索路径中以更新用户的 **PATH**: 在实例所有者主目录的 `sqllib` 子目录下的 `bin`、`adm` 和 `misc` 子目录。
- 将 **DB2INSTANCE** 环境变量设置为实例名。

## 实例目录

实例目录存储着与一个数据库实例相关的所有信息。实例目录一旦创建，就不能更改其位置。

实例目录包含:

- 数据库管理器配置文件
- 系统数据库目录

- 节点目录
- 节点配置文件（db2nodes.cfg）
- 包含调试信息（例如，异常或寄存器转储或用于 DB2 进程的调用堆栈）的其他文件。

在 Linux 和 UNIX 操作系统上，实例目录位于 INSTHOME/sqllib 目录中，其中 INSTHOME 是实例所有者的主目录。只要符合命名规则，就可以随意命名缺省实例。

在 Windows 操作系统上，实例目录位于安装了 DB2 数据库产品的 /sqllib 目录下。实例名与服务名称相同，因此应该不会发生冲突。实例名不应与别的服务名称相同。您必须要有创建服务所需的正确权限。

在分区数据库环境中，该实例目录是由属于该实例的所有数据库分区服务器共享的。因此，必须在该实例中的所有计算机可以访问的一个网络共享驱动器上创建实例目录。

### db2nodes.cfg

db2nodes.cfg 文件用来定义参与 DB2 实例的数据库分区服务器。如果想要将高速互连用于数据库分区服务器通信，那么还可以使用 db2nodes.cfg 文件来指定高速互连的 IP 地址或主机名。

## 多个实例（Linux 和 UNIX）

如果 DB2 产品是由具有 root 用户特权的用户安装的，那么在 Linux 或 UNIX 操作系统上可以有多个实例。虽然每个实例同时运行，但每个实例都是独立的。因此，每次只能在数据库管理器的一个实例内工作。

**注:** 要防止两个或多个实例之间的环境冲突，应确保每个实例都有它自己的主目录。如果共享主目录，那么会返回错误。每个主目录可以位于相同或不同文件系统中。

实例所有者和“系统管理”（SYSADM）组与每个实例相关。实例所有者和 SYSADM 组是在创建实例期间指定的。一个用户标识或用户名只能用于一个实例，并且该用户标识或用户名也称为*实例所有者*。

每个实例所有者必须有一个唯一的主目录。运行实例需要的所有配置文件都在该实例所有者的用户标识或用户名的主目录中创建。如果需要从系统中除去实例所有者的用户标识或用户名，可能会丢失与该实例相关的文件并失去对存储在此实例中的数据的访问权。因此，将要独占使用的实例所有者用户标识或用户名专用于运行数据库管理器。

实例所有者的主组也很重要。此主组自动成为该实例的系统管理组，并获得该实例的 SYSADM 权限。作为该实例所有者主组成员的其他用户标识或用户名也获得此级别的权限。因此，可能想要将该实例所有者的用户标识或用户名指定给为管理实例而保留的一个主组。（还要确保将一个主组指定给该实例所有者的用户标识或用户名；否则，使用系统的缺省主组。）

如果已经有一个组并希望使它成为该实例的系统管理组，当创建实例所有者的用户标识或用户名时可将此组简单指定为主组。要赋予其他用户对该实例的管理权限，将他们添加到指定作为系统管理组的那一组。

为了将不同实例的 SYSADM 权限区别开，确保每个实例所有者的用户标识或用户名使用不同的主组。但是，如果选择对多个实例具有公共的 SYSADM 权限，那么可对这多个实例使用同一个主组。

## 多个实例（Windows）

可以在同一台计算机上运行 DB2 数据库管理器的多个实例。数据库管理器的每个实例维护其自己的数据库且具有自己的数据库管理器配置参数。

**注**: 实例也可以属于计算机上处于不同数据库管理器级别的不同 DB2 副本。如果您正在运行 64 位 Windows 系统，那么可安装 32 位 DB2 或 64 位 DB2，但这两者不能在同一机器上共存。

数据库管理器实例由下列内容组成:

- 表示该实例的 Windows 服务。服务的名称与实例名相同。服务的显示名称（在**服务**面板中）是实例名加上“DB2 -”字符串前缀。例如，对于名为“DB2”的实例，存在名为“DB2”并且显示名为“DB2 - *DB2 副本名* - DB2”的 Windows 服务。

  **注**: 不会为客户机实例创建 Windows 服务。
- 实例目录。此目录包含数据库管理器配置文件、系统数据库目录、节点目录、数据库连接服务（DCS）目录以及与实例相关联的所有诊断日志和转储文件。实例目录随 Windows 操作系统系列的版本不同而不同；要验证 Windows 上的缺省目录，请使用 `db2set DB2INSTPROF` 命令来检查 **DB2INSTPROF** 环境变量的设置。您还可以通过更改 **DB2INSTPROF** 环境变量来更改缺省实例目录。例如，将其设置为 c:\DB2PROFS:
  - 使用 db2set.exe **-g** 命令将 **DB2INSTPROF** 设置为 c:\DB2PROFS
  - 运行 DB2ICRT.exe 命令来创建此实例。
- 在 Windows 操作系统上创建实例时，用户数据文件（例如，实例目录和 `db2cli.ini` 文件）的缺省位置为下列目录:
  - 在 Windows XP 和 Windows 2003 操作系统上: `Documents and Settings\All Users\Application Data\IBM\DB2\`*`Copy Name`*
  - 在 Windows 2008 和 Windows Vista（和更高版本）操作系统上: `Program Data\IBM\DB2\`*`Copy Name`*

  其中，*Copy Name* 表示 DB2 副本名。

**注**: `db2cli.ini` 文件的位置可能会根据是否使用了 Microsoft ODBC 驱动程序管理器、所使用的数据源名称（DSN）的类型、所安装的客户机或驱动程序的类型以及是否设置了注册表变量 **DB2CLIINIPATH** 而不同。有关更多信息，请参阅 *Call Level Interface Guide and Reference, Volume 1* 中的“db2cli.ini initialization file”主题。

# 创建实例

虽然实例是作为数据库管理器安装的一部分创建的，但业务需求可能需要创建其他实例。

**先决条件**

如果您在 Windows 上属于“Administrators”组，或者您在 Linux 或 UNIX 平台上具有 root 用户权限，那么您可以添加其他实例。添加实例的计算机将成为“实例拥有的计算机”（节点零）。一定要在 DB2 管理服务器所在的计算机上添加实例。实例标识不应为 root 或者具有已到期的密码。

**限制**

- 在 Linux 和 UNIX 操作系统上，不能为非 root 安装创建更多实例。
- 如果使用现有用户标识来创建 DB2 实例，那么要确保该用户标识:
  - 未被锁定
  - 密码未到期

要使用命令行添加实例，请输入:

```
db2icrt <instance_name>
```

在 AIX 服务器上创建实例时，必须提供受防护的用户标识，例如:

```
DB2DIR/instance/db2icrt -u db2fenc1 db2inst1
```

使用 db2icrt 命令来添加另一个 DB2 实例时，应提供实例所有者的登录名以及（可选）指定该实例的认证类型。该认证类型适用于在该实例下创建的所有数据库。认证类型是对将在何处进行用户认证的说明。

可在 DB2PATH 中使用 DB2INSTPROF 环境变量更改实例目录的位置。需要该实例目录的写访问权。如果想要在不同于 DB2PATH 的路径中创建目录，那么输入 db2icrt 命令之前必须设置 DB2INSTPROF。

对于 DB2 Enterprise Server Edition（ESE），还需要声明您正在添加的新实例是分区数据库系统。另外，使用带有多个数据库分区的 ESE 实例，并使用“快速通信管理器”（FCM）时，通过在创建实例时定义更多 TCP/IP 端口，可以在数据库分区间建立多个连接。

例如，对于 Windows 操作系统，使用带有 -r <port range> 参数的 db2icrt 命令。端口范围按如下所示显示，其中 base_port 是 FCM 可使用的第一个端口，而 end_port 是 FCM 可使用的端口范围内的最后一个端口:

```
-r:<base_port,end_port>
```

## 修改实例

实例都设计为尽可能与以后产品的安装和除去所产生的影响无关。在 Linux 和 UNIX 上，可以在安装或除去可执行文件或组件后更新实例。在 Windows 上，运行 db2iupdt 命令。

在大多数情况下，现有实例自动继承或失去要安装或除去的产品功能的访问权。但是，如果安装或除去了特定的可执行文件或组件，那么现有实例不会自动继承新的系统配置参数或获得所有其他功能的访问权。必须更新该实例。

如果通过安装“程序临时性修订”（PTF）或补丁更新了数据库管理器，那么应使用 db2iupdt 命令（root 用户安装）或 db2nrupdt 命令（非 root 用户安装）来更新所有现有数据库实例。

在尝试更改或删除实例前，应确保了解那些实例和实例中已有的数据库分区服务器。

## 更新实例配置（Linux 和 UNIX）

本主题仅适用于根实例。要更新非根实例，请运行 db2nrupdt 命令。

运行 db2iupdt 命令，并执行以下操作来更新指定的实例:

- 替换实例所有者主目录下 sqllib 子目录中的文件。
- 如果更改了节点类型，那么会创建一个新的数据库管理器配置文件。为此，可将现有的数据库管理器配置文件的相关值与新节点类型的缺省数据库管理器配置文件合并。如果创建了一个新的数据库管理器配置文件，那么将旧文件备份到实例所有者主目录下的 sqllib 子目录的 backup 子目录中。

db2iupdt 命令可在 AIX 上的 usr/opt/db2_09_05/instance/ 目录中找到。db2iupdt 命令可在 HP-UX、Solaris 或 Linux 上的 /opt/IBM/db2/V9.5/instance 目录中找到。

要使用命令行更新实例，请输入:

```
db2iupdt InstName
```

InstName 是实例所有者的登录名。

此命令还有其他可选参数:

**-h 或 -?**
显示此命令的帮助菜单。

**-d** 设置在问题确定期间要使用的调试方式。

**-a AuthType**
指定实例的认证类型。有效的认证类型是 SERVER、SERVER_ENCRYPT 或 CLIENT。如果未指定此参数，且已安装了 DB2 服务器，那么缺省值为 SERVER。否则，将它设置为 CLIENT。该实例的认证类型适用于实例拥有的所有数据库。

**-e** 允许更新存在的每个实例。使用 db2ilist 来列示现有实例。

**-u Fenced ID**
命名受防护的用户定义的函数（UDF）和存储过程执行期间所归属的用户。如果安装了数据服务器客户机或 DB2 软件开发者工具箱，那么不需要这样做。对于其他 DB2 产品，这是必需参数。注意: Fenced ID 不能是“root”或“bin”。

**-k** 此参数保留当前实例类型。若不指定此参数，当前实例将按以下顺序升级到可用的最高实例类型:

- 带有本地和远程客户机的分区数据库服务器
- 带有本地和远程客户机的数据库服务器
- 客户机

示例:

- 如果在创建实例后安装了 DB2 Workgroup Server Edition或 DB2 Enterprise Server Edition，可输入以下命令来更新该实例:

  ```
  db2iupdt -u db2fenc1 db2inst1
  ```

- 如果在创建实例后安装了 DB2 Connect™ Enterprise Edition，那么也可以将实例名用作 Fenced ID:

  ```
  db2iupdt -u db2inst1 db2inst1
  ```

- 要更新客户机实例，请调用以下命令:

```
db2iupdt db2inst1
```

## 更新实例配置（Windows）

要更新 Windows 上的实例配置，请使用 db2iupdt 命令。

运行 db2iupdt 命令，并执行以下操作来更新指定的实例:

- 替换实例所有者主目录下 sqllib 子目录中的文件。
- 如果更改了节点类型，那么会创建一个新的数据库管理器配置文件。为此，可将现有的数据库管理器配置文件的相关值与新节点类型的缺省数据库管理器配置文件合并。如果创建了一个新的数据库管理器配置文件，那么将旧文件备份到实例所有者主目录下的 sqllib 子目录的 backup 子目录中。

db2iupdt 命令可在 \sqllib\bin 目录中找到。

按如下所示使用该命令:

```
db2iupdt InstName
```

InstName 是实例所有者的登录名。

此命令还有其他可选参数:

**/h: hostname**
覆盖缺省 TCP/IP 主机名（如果当前计算机有一个或多个 TCP/IP 主机名）。

**/p: instance profile path**
为已更新实例指定新的实例概要文件路径。

**/r: baseport,endport**
指定在运行多个数据库分区时，分区数据库实例将使用的 TCP/IP 端口范围。

**/u: username,password**
指定 DB2 服务的帐户和密码。

# 使用实例

使用实例时，可以启动或停止实例，并将它连接至实例或从实例拆离。

每个实例由属于在*实例配置文件*（也称为*数据库管理器配置文件*）中定义的 SYSADM_GROUP 的用户来管理。对于每个操作环境来说，创建用户标识和用户组都不相同。

## 自动启动实例

在 Windows 操作系统上，缺省情况下，安装期间创建的数据库实例设置为自动启动。使用 db2icrt 创建的实例设置为手动启动。要更改启动类型，需要转至“服务”面板并在其中更改 DB2 服务的属性。

在 UNIX 操作系统上，要允许一个实例在每次系统重新启动后自动启动，请输入以下命令:

```
db2iauto -on <instance name>
```

其中 <instance name> 是实例的登录名。在 UNIX 操作系统上，要阻止一个实例在每次系统重新启动后自动启动，请输入以下命令:

```
db2iauto -off <instance name>
```

其中 <instance name> 是实例的登录名。

## 启动实例（Linux 和 UNIX）

在正常业务操作期间，可能需要启动或停止 DB2 数据库；例如，必须启动一个实例，然后才能执行下列某些任务：连接至该实例中的数据库、预编译应用程序、将程序包绑定至数据库或访问主机数据库。

在 Linux 或 UNIX 系统上启动实例之前:

1. 使用对该实例具有 SYSADM、SYSCTRL 或 SYSMAINT 权限的用户标识或名称进行登录；或者以实例所有者身份登录。
2. 按如下所示运行启动脚本，其中 INSTHOME 是要使用的实例的主目录:

```
. INSTHOME/sqllib/db2profile      （对于 Bourne 或 Korn shell 程序）
  source INSTHOME/sqllib/db2cshrc   （对于 C shell）
```

使用命令行来启动实例，请输入:

```
db2start
```

**注**: 运行命令以启动或停止实例的数据库管理器时，DB2 数据库管理器将该命令应用于当前实例。

## 启动实例（Windows）

在正常业务操作期间，可能需要启动或停止 DB2 实例；例如，必须启动一个实例，然后才能执行下列某些任务：连接至该实例中的数据库、预编译应用程序、将程序包绑定至数据库或访问主机数据库。

要通过 db2start 将 DB2 数据库实例作为服务成功启动，用户帐户必须具有 Windows 操作系统定义的、用于启动 Windows 服务的正确特权。用户帐户可以是 Administrators、Server Operators 或 Power Users 组的一个成员。启用了扩展安全性之后，缺省情况下，只有 DB2ADMNS 和 Administrators 组的成员才能启动数据库。

要使用命令行启动实例，请输入:

```
db2start
```

**注**: 运行命令以启动或停止实例的数据库管理器时，DB2 数据库管理器将该命令应用于当前实例。

db2start 命令将 DB2 数据库实例作为 Windows 服务来启动。通过在调用 db2start 时指定“/D”开关，仍可以在 Windows 上将 DB2 数据库实例作为进程运行。还可使用“控制面板”或 NET START 命令将 DB2 数据库实例作为服务启动。

当在分区数据库环境中运行时，每个数据库分区服务器都是作为 Windows 服务启动的。在分区数据库环境中，不能使用“/D”开关将DB2 实例作为进程启动。

## 连接至实例和从实例拆离

在所有平台上，要与另一个可能是远程的数据库管理器的实例连接，请使用 ATTACH 命令。要从实例拆离，请使用 DETACH 命令。

必须存在多个实例。

要使用命令行来与实例连接，请输入:

```
db2 attach to <instance name>
```

例如，要连接至节点目录中先前编目的称为 testdb2 的实例:

```
db2 attach to testdb2
```

例如，在对 testdb2 实例执行维护活动后，要使用命令行从实例拆离，请输入:

```
db2 detach
```

### 连接至客户机应用程序和从客户机应用程序拆离

- 要连接至客户机应用程序中的实例，请调用 sqleatin API。
- 要与客户机应用程序中的实例拆离，请调用 sqledtin API。

## 使用同一 DB2 副本或不同 DB2 副本上的实例

可以在同一 DB2 副本或不同 DB2 副本中同时运行多个实例。

要使用同一 DB2 副本中的多个实例，您需要:

1. 创建所有实例或将它们迁移至同一 DB2 副本。
2. 在对要使用的实例发出命令之前，将 DB2INSTANCE 环境变量设置为该实例的名称。

要阻止实例访问另一实例的数据库，可在与实例同名的目录下为实例创建数据库文件。例如，在驱动器 C: 上为实例“DB2”上创建数据库时，会在称为 C:\DB2 的目录中创建数据库文件。类似地，在驱动器 C: 上为实例 TEST 创建数据库时，会在称为 C:\TEST 的目录中创建数据库文件。缺省情况下，它的值为安装了 DB2 产品的盘符。有关更多信息，请参阅 *dftdbpath* 数据库管理器配置参数。

要在具有多个 DB2 副本的系统中使用实例，请使用下列任一方法:

- 通过以下途径使用命令窗口: 选择“开始”→“程序”→“IBM DB2”→ *<DB2 副本名称>* →“命令行工具”→“命令窗口”。已使用选择的特定 DB2 副本的正确环境变量设置该命令窗口。
- 从命令窗口中使用 db2envar.bat:
  1. 打开命令窗口。
  2. 通过使用想要应用程序使用的 DB2 副本的标准路径来运行 db2envar.bat 文件:

     ```
     <DB2 Copy install dir>\bin\db2envar.bat
     ```

## 停止实例（Linux 和 UNIX）

您可能需要停止数据库管理器的当前实例。

### 开始之前

要在 Linux 或 UNIX 系统上停止实例，必须执行下列操作:

1. 使用对实例具有 SYSADM、SYSCTRL 或 SYSMAINT 权限的用户标识或名称登录或连接至实例；或者以实例所有者身份登录。
2. 显示与要停止的特定数据库连接的所有应用程序和用户。要确保没有关键性的或极重要的应用程序在运行，列示应用程序。为此，需要 SYSADM、SYSCTRL 或 SYSMAINT 权限。
3. 强制所有应用程序和用户与该数据库断开。需要 SYSADM 或 SYSCTRL 权限来强制用户。

### 关于本任务

db2stop 命令只能在服务器上运行。当运行此命令时，不允许有任何数据库连接；但是，如果有任何实例连接，那么在停止实例之前要将其强制断开。

**注**: 如果命令行处理器会话与一个实例连接，那么必须在运行 db2stop 命令前运行 terminate 命令来结束每个会话。db2stop 命令将停止由 **DB2INSTANCE** 环境变量定义的实例。

**注**: 运行命令以启动或停止实例的数据库管理器时，DB2 数据库管理器将该命令应用于当前实例。有关更多信息，请参阅设置当前实例环境变量。

### 过程

要使用命令行来停止实例，请输入: `db2stop`
可以使用 db2stop 命令来停止或删除分区数据库环境中的各个数据库分区。当在分区数据库环境中工作时，试图使用以下命令删除逻辑分区

```
db2stop drop nodenum <0>
```

必须确保没有用户在试图访问该数据库。如果有，那么将接收到错误消息 SQL6030N。

## 停止实例（Windows）

您可能需要停止数据库管理器的当前实例。

### 开始之前

要在系统上停止实例，必须执行下列操作:

1. 停止 DB2 数据库服务的用户帐户必须具有 Windows操作系统定义的正确特权。用户帐户可以是 Administrators、Server Operators 或 Power Users 组的一个成员。
2. 显示与要停止的特定数据库连接的所有应用程序和用户。要确保没有关键性的或极重要的应用程序在运行，列示应用程序。为此，需要 SYSADM、SYSCTRL 或 SYSMAINT 权限。
3. 强制所有应用程序和用户与该数据库断开。需要 SYSADM 或 SYSCTRL 权限来强制用户。

### 关于本任务

db2stop 命令只能在服务器上运行。当运行此命令时，不允许任何数据库连接；但是，如果有任何实例连接，那么在停止 DB2 数据库服务前要将它们强制断开。

**注:** 如果命令行处理器会话与一个实例连接，那么必须在运行 db2stop 命令前运行 terminate 命令来结束每个会话。db2stop 命令将停止由 **DB2INSTANCE** 环境变量定义的实例。

请记住，在分区数据库环境中使用数据库管理器时，每个数据库分区服务器都将作为服务启动。必须停止每个服务。

**注:** 当运行命令以启动或停止实例的数据库管理器时，数据库管理器将该命令应用于当前实例。有关更多信息，请参阅设置当前实例环境变量。

### 过程

要在系统上停止实例，使用下列其中一个方法:

- 使用 db2stop 命令停止。
- 使用 NET STOP 命令停止。
- 从应用程序中停止实例。

## 删除实例

要删除根实例，请发出 db2idrop 命令。要删除非根实例，必须卸载 DB2 数据库产品。

### 过程

要使用命令行来除去根实例:

1. 停止当前使用该实例的所有应用程序。
2. 在每个 DB2 命令窗口中，通过运行 db2 terminate 命令来停止命令行处理器。
3. 运行 db2stop 命令来停止该实例。
4. 备份由 **DB2INSTPROF** 注册表变量指示的实例目录。

   在 Linux 和 UNIX 操作系统上，请考虑备份 *INSTHOME*/sqllib 目录中的文件（其中 *INSTHOME* 是实例所有者的主目录）。例如，可能想保存数据库管理器配置文件 db2systm、db2nodes.cfg 文件、用户定义的函数（UDF）或受防护的存储过程应用程序。
5. （仅限于 Linux 和 UNIX 操作系统）作为实例所有者注销，然后作为以 root 用户权限的用户登录。
6. 发出 db2idrop 命令，如下所示:

   ```
   db2idrop InstName
   ```

   其中 *InstName* 是要删除的实例的名称。

   db2idrop 命令从实例列表中除去实例条目，并除去实例所有者主目录下的 sqllib 子目录。

   **注:** 在 Linux 和 UNIX 操作系统上，如果您发出 db2idrop 命令并且接收到一条消息，指出无法除去 *INSTHOME*/sqllib 子目录，那么其中一个原因可能是 *INSTHOME*/adm 子目录中包含具有 .nfs 扩展名的文件。adm 子目录是安装了 NFS 的系统，而这些文件在服务器上是受控的。必须从安装目录的文件服务器中删除 *.nfs 文件。然后可除去 *INSTHOME*/sqllib 子目录。

7. 对于 Windows 操作系统，如果您已删除的实例是缺省实例，那么通过发出 db2set 命令来设置新的缺省实例:

   ```
   db2set db2instdef=instance_name -g
   ```

   其中 *instance_name* 是现有实例的名称。

8. 对于 Linux 和 UNIX 操作系统，除去该实例所有者的用户标识和组（如果仅用于该实例）。 如果您打算重新创建实例，那么不要除去这些内容。

   此步骤是可选的，因为实例所有者和实例所有者组可用于其他用途。

# 第 5 章 轻量级目录访问协议（LDAP）

“轻量级目录访问协议”（LDAP）是一种用于访问目录服务的业界标准方法。目录服务是一个关于分布式环境中的多个系统和服务的资源信息的存储库；它提供对这些资源的客户机和服务器访问。

每个数据库服务器实例都将它的存在情况发布给 LDAP 服务器，并在创建数据库时向 LDAP 目录提供数据库信息。客户机与数据库连接后，就可以从 LDAP 目录检索服务器的目录信息。不再要求每个客户机将目录信息以本地方式存储在每台机器上。客户机应用程序搜索 LDAP 目录以找出连接数据库所需的信息。

由于存在高速缓存机制，因此客户机仅需要搜索 LDAP 目录服务器一次。从 LDAP 目录服务器中检索到信息之后，会根据数据库管理器配置参数 **dir_cache** 和注册表变量 **DB2LDAPCACHE** 的值，在本地机器上存储或高速缓存此信息。**dir_cache** 数据库管理器配置参数用于在内存高速缓存中存储数据库、节点和 DCS 目录文件。应用程序使用目录高速缓存直至应用程序关闭。**DB2LDAPCACHE** 注册表变量用于在本地磁盘高速缓存中存储数据库、节点和 DCS 目录文件。

- 如果 **DB2LDAPCACHE**=NO 且 **dir_cache**=NO，那么始终从 LDAP 中读取信息。
- 如果 **DB2LDAPCACHE**=NO 且 **dir_cache**=YES，那么从 LDAP 中读取一次信息，并将其插入到 DB2 高速缓存中。
- 如果 **DB2LDAPCACHE**=YES 或者未设置此注册表变量，那么从 LDAP 中读取一次信息，并将它高速缓存到本地数据库、节点和 DCS 目录中。

**注**：**DB2LDAPCACHE** 注册表变量仅适用于数据库和节点目录。

## LDAP 环境中的安全性注意事项

在访问 LDAP 目录中的信息之前，LDAP 服务器要认证应用程序或用户。认证过程已*绑定*至 LDAP 服务器。对存储在 LDAP 目录中的信息进行访问控制很重要，这样可以防止匿名用户添加、删除或修改信息。

缺省情况下，将继承访问控制，并可在容器级应用。当创建了一个新对象时，它就继承父对象的相同的安全性属性。可使用 LDAP 服务器的管理工具来定义容器对象的访问控制。

缺省情况下，按如下所示定义访问控制:

- 对于 LDAP 中的数据库条目和节点条目，每个人（或任何匿名用户）都有读取权限。只有目录管理员以及对象的所有者或创建者具有读/写访问权。
- 对于用户概要文件，概要文件所有者和目录管理员具有读/写访问权。如果一个用户没有“目录管理员”权限，那么不能访问另一个用户的概要文件。

**注**：权限检查始终由 LDAP 服务器执行，而不是由 DB2 执行。LDAP 权限检查与 DB2 权限无关。具有 SYSADM 权限的帐户或授权标识也许不能访问 LDAP 目录。

© Copyright IBM Corp. 1993, 2010

当运行 LDAP 命令或 API 时，如果未指定绑定“专有名称”（bindDN）和密码，那么 DB2 使用缺省凭证绑定至 LDAP 服务器，该凭证可能没有足够的权限来执行请求的命令，将返回错误。

可使用 DB2 命令或 API 的 USER 和 PASSWORD 子句显式地指定用户的 bindDN 和密码。

## DB2 使用的 LDAP 对象类和属性

下面的表描述了 DB2 数据库管理器使用的对象类:

*表 10. cimManagedElement*

| 类 | cimManagedElement |
|---|---|
| Active Directory LDAP 显示名称 | 不适用 |
| Active Directory 公共名（cn） | 不适用 |
| 描述 | 提供“IBM 模式”中许多系统管理对象类的基类 |
| SubClassOf | 顶部 |
| 必需的属性 | |
| 可选属性 | 描述 |
| 类型 | 抽象 |
| OID（对象标识） | 1.3.18.0.2.6.132 |
| GUID（全局唯一标识） | b3afd63f-5c5b-11d3-b818-002035559151 |

*表 11. cimSetting*

| 类 | cimSetting |
|---|---|
| Active Directory LDAP 显示名称 | 不适用 |
| Active Directory 公共名（cn） | 不适用 |
| 描述 | 提供“IBM 模式”中的配置和设置的基类 |
| SubClassOf | cimManagedElement |
| 必需的属性 | |
| 可选属性 | settingID |
| 类型 | 抽象 |
| OID（对象标识） | 1.3.18.0.2.6.131 |
| GUID（全局唯一标识） | b3afd64d-5c5b-11d3-b818-002035559151 |

*表 12. eProperty*

| 类 | eProperty |
|---|---|
| Active Directory LDAP 显示名称 | ibm-eProperty |
| Active Directory 公共名（cn） | ibm-eProperty |
| 描述 | 用来指定用户首选项属性的任何特定于应用程序的设置 |
| SubClassOf | cimSetting |
| 必需的属性 | |

表 12. *eProperty（续）*

| 类 | eProperty |
| --- | --- |
| 可选属性 | propertyType<br>cisPropertyType<br>cisProperty<br>cesPropertyType<br>cesProperty<br>binPropertyType<br>binProperty |
| 类型 | 结构 |
| OID（对象标识） | 1.3.18.0.2.6.90 |
| GUID（全局唯一标识） | b3afd69c-5c5b-11d3-b818-002035559151 |

表 13. *DB2Node*

| 类 | DB2Node |
| --- | --- |
| Active Directory LDAP 显示名称 | ibm-db2Node |
| Active Directory 公共名（cn） | ibm-db2Node |
| 描述 | 表示 DB2 服务器 |
| SubClassOf | eSap / ServiceConnectionPoint |
| 必需的属性 | db2nodeName |
| 可选属性 | db2nodeAlias<br>db2instanceName<br>db2Type<br>host / dNSHostName（请参阅注释 2）<br>protocolInformation/ServiceBindingInformation |
| 类型 | 结构 |
| OID（对象标识） | 1.3.18.0.2.6.116 |
| GUID（全局唯一标识） | b3afd65a-5c5b-11d3-b818-002035559151 |
| 特殊Notes® | 1. *DB2Node* 类是从 IBM Tivoli® Directory Server 下面的 *eSap* 对象类和 Microsoft Active Directory 下面的 *ServiceConnectionPoint* 对象类派生而来的。<br>2. *host* 在 IBM Tivoli Directory Server 环境中使用。*dNSHostName* 属性用于 Microsoft Active Directory 中。<br>3. *protocolInformation* 仅在 IBM Tivoli Directory Server 环境中使用。对于 Microsoft Active Directory，从 ServiceConnectionPoint 类继承的 *ServiceBindingInformation* 属性用来包含协议信息。 |

*DB2Node* 对象中的 *protocolInformation*（在 IBM Tivoli Directory Server 中）或 *ServiceBindingInformation*（在 Microsoft Active Directory 中）属性包含用来绑定 DB2 数据库服务器的通信协议信息。它由标记组成，这些标记描述受支持的网络协议。标记之间由分号隔开。标记之间没有空格。可使用星号（*）来指定可选参数。

TCP/IP 的标记为:

- “TCPIP”
- 服务器主机名或 IP 地址
- 服务名称（svcename）或端口号（例如，50000）
- （可选）安全性（“NONE”或“SOCKS”）

“命名管道”的标记为:

- “NPIPE”
- 服务器的计算机名称
- 服务器的实例名

*表 14. DB2Database*

| **类** | **DB2Database** |
|---|---|
| Active Directory LDAP 显示名称 | ibm-db2Database |
| Active Directory 公共名（cn） | ibm-db2Database |
| 描述 | 表示 DB2 数据库 |
| SubClassOf | 顶部 |
| 必需的属性 | db2databaseName<br><br>db2nodePtr |
| 可选属性 | db2databaseAlias<br><br>db2additionalParameters<br><br>db2ARLibrary<br><br>db2authenticationLocation<br><br>db2gwPtr<br><br>db2databaseRelease<br><br>DCEPrincipalName<br><br>db2altgwPtr<br><br>db2altnodePtr |
| 类型 | 结构 |
| OID（对象标识） | 1.3.18.0.2.6.117 |
| GUID（全局唯一标识） | b3afd659-5c5b-11d3-b818-002035559151 |

*表 15. db2additionalParameters*

| **属性** | **db2additionalParameters** |
|---|---|
| Active Directory LDAP 显示名称 | ibm-db2AdditionalParameters |

表 15. *db2additionalParameters (续)*

| 属性 | db2additionalParameters |
|---|---|
| Active Directory 公共名（cn） | ibm-db2AdditionalParameters |
| 描述 | 包含连接主机数据库服务器时使用的附加参数 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 1024 |
| 多值 | 单值 |
| OID（对象标识） | 1.3.18.0.2.4.426 |
| GUID（全局唯一标识） | b3afd315-5c5b-11d3-b818-002035559151 |

表 16. *db2authenticationLocation*

| 属性 | db2authenticationLocation |
|---|---|
| Active Directory LDAP 显示名称 | ibm-db2AuthenticationLocation |
| Active Directory 公共名（cn） | ibm-db2AuthenticationLocation |
| 描述 | 指定执行认证的位置 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 64 |
| 多值 | 单值 |
| OID（对象标识） | 1.3.18.0.2.4.425 |
| GUID（全局唯一标识） | b3afd317-5c5b-11d3-b818-002035559151 |
| 注意事项 | 有效值有：<br>CLIENT、SERVER、DCS、DCE、KERBEROS、SVRENCRYPT 或 DCSENCRYPT |

表 17. *db2ARLibrary*

| 属性 | db2ARLibrary |
|---|---|
| Active Directory LDAP 显示名称 | ibm-db2ARLibrary |
| Active Directory 公共名（cn） | ibm-db2ARLibrary |
| 描述 | “应用程序请求程序”库的名称 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 256 |
| 多值 | 单值 |
| OID（对象标识） | 1.3.18.0.2.4.427 |
| GUID（全局唯一标识） | b3afd316-5c5b-11d3-b818-002035559151 |

表 18. *db2databaseAlias*

| 属性 | db2databaseAlias |
|---|---|
| Active Directory LDAP 显示名称 | ibm-db2DatabaseAlias |
| Active Directory 公共名（cn） | ibm-db2DatabaseAlias |
| 描述 | 数据库别名 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 1024 |

表 18. db2databaseAlias（续）

| 属性 | db2databaseAlias |
| --- | --- |
| 多值 | 多值 |
| OID（对象标识） | 1.3.18.0.2.4.422 |
| GUID（全局唯一标识） | b3afd318-5c5b-11d3-b818-002035559151 |

表 19. db2databaseName

| 属性 | db2databaseName |
| --- | --- |
| Active Directory LDAP 显示名称 | ibm-db2DatabaseName |
| Active Directory 公共名（cn） | ibm-db2DatabaseName |
| 描述 | 数据库名称 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 1024 |
| 多值 | 单值 |
| OID（对象标识） | 1.3.18.0.2.4.421 |
| GUID（全局唯一标识） | b3afd319-5c5b-11d3-b818-002035559151 |

表 20. db2databaseRelease

| 属性 | db2databaseRelease |
| --- | --- |
| Active Directory LDAP 显示名称 | ibm-db2DatabaseRelease |
| Active Directory 公共名（cn） | ibm-db2DatabaseRelease |
| 描述 | 数据库发行版号 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 64 |
| 多值 | 单值 |
| OID（对象标识） | 1.3.18.0.2.4.429 |
| GUID（全局唯一标识） | b3afd31a-5c5b-11d3-b818-002035559151 |

表 21. db2nodeAlias

| 属性 | db2nodeAlias |
| --- | --- |
| Active Directory LDAP 显示名称 | ibm-db2NodeAlias |
| Active Directory 公共名（cn） | ibm-db2NodeAlias |
| 描述 | 节点别名 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 1024 |
| 多值 | 多值 |
| OID（对象标识） | 1.3.18.0.2.4.420 |
| GUID（全局唯一标识） | b3afd31d-5c5b-11d3-b818-002035559151 |

表 22. db2nodeName

| 属性 | db2nodeName |
| --- | --- |
| Active Directory LDAP 显示名称 | ibm-db2NodeName |

表 22. *db2nodeName（续）*

| 属性 | db2nodeName |
|---|---|
| Active Directory 公共名（cn） | ibm-db2NodeName |
| 描述 | 节点名 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 64 |
| 多值 | 单值 |
| OID（对象标识） | 1.3.18.0.2.4.419 |
| GUID（全局唯一标识） | b3afd31e-5c5b-11d3-b818-002035559151 |

表 23. *db2nodePtr*

| 属性 | db2nodePtr |
|---|---|
| Active Directory LDAP 显示名称 | ibm-db2NodePtr |
| Active Directory 公共名（cn） | ibm-db2NodePtr |
| 描述 | 指向表示拥有数据库的数据库服务器的“节点”（DB2Node）对象的指针 |
| 语法 | 专有名称 |
| 最大长度 | 1000 |
| 多值 | 单值 |
| OID（对象标识） | 1.3.18.0.2.4.423 |
| GUID（全局唯一标识） | b3afd31f-5c5b-11d3-b818-002035559151 |
| 特殊注意事项 | 此关系允许客户机检索用来连接数据库的协议通信信息。 |

表 24. *db2altnodePtr*

| 属性 | db2altnodePtr |
|---|---|
| Active Directory LDAP 显示名称 | ibm-db2AltNodePtr |
| Active Directory 公共名（cn） | ibm-db2AltNodePtr |
| 描述 | 指向表示备用数据库服务器的“节点”（DB2Node）对象的指针 |
| 语法 | 专有名称 |
| 最大长度 | 1000 |
| 多值 | 多值 |
| OID（对象标识） | 1.3.18.0.2.4.3093 |
| GUID（全局唯一标识） | 5694e266-2059-4e32-971e-0778909e0e72 |

表 25. *db2gwPtr*

| 属性 | db2gwPtr |
|---|---|
| Active Directory LDAP 显示名称 | ibm-db2GwPtr |
| Active Directory 公共名（cn） | ibm-db2GwPtr |
| 描述 | 指向表示网关服务器的“节点”对象的指针，可从该网关服务器访问数据库 |
| 语法 | 专有名称 |

*表 25. db2gwPtr (续)*

| 属性 | db2gwPtr |
|---|---|
| 最大长度 | 1000 |
| 多值 | 单值 |
| OID（对象标识） | 1.3.18.0.2.4.424 |
| GUID（全局唯一标识） | b3afd31b-5c5b-11d3-b818-002035559151 |

*表 26. db2altgwPtr*

| 属性 | db2altgwPtr |
|---|---|
| Active Directory LDAP 显示名称 | ibm-db2AltGwPtr |
| Active Directory 公共名（cn） | ibm-db2AltGwPtr |
| 描述 | 指向表示备用网关服务器的“节点”对象的指针 |
| 语法 | 专有名称 |
| 最大长度 | 1000 |
| 多值 | 多值 |
| OID（对象标识） | 1.3.18.0.2.4.3092 |
| GUID（全局唯一标识） | 70ab425d-65cc-4d7f-91d8-084888b3a6db |

*表 27. db2instanceName*

| 属性 | db2instanceName |
|---|---|
| Active Directory LDAP 显示名称 | ibm-db2InstanceName |
| Active Directory 公共名（cn） | ibm-db2InstanceName |
| 描述 | 数据库服务器实例的名称 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 256 |
| 多值 | 单值 |
| OID（对象标识） | 1.3.18.0.2.4.428 |
| GUID（全局唯一标识） | b3afd31c-5c5b-11d3-b818-002035559151 |

*表 28. db2Type*

| 属性 | db2Type |
|---|---|
| Active Directory LDAP 显示名称 | ibm-db2Type |
| Active Directory 公共名（cn） | ibm-db2Type |
| 描述 | 数据库服务器的类型 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 64 |
| 多值 | 单值 |
| OID（对象标识） | 1.3.18.0.2.4.418 |
| GUID（全局唯一标识） | b3afd320-5c5b-11d3-b818-002035559151 |
| 注意事项 | 数据库服务器的有效类型是：SERVER、MPP 和 DCS |

表 29. *DCEPrincipalName*

| 属性 | DCEPrincipalName |
|---|---|
| Active Directory LDAP 显示名称 | ibm-DCEPrincipalName |
| Active Directory 公共名（cn） | ibm-DCEPrincipalName |
| 描述 | DCE 主体名称 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 2048 |
| 多值 | 单值 |
| OID（对象标识） | 1.3.18.0.2.4.443 |
| GUID（全局唯一标识） | b3afd32d-5c5b-11d3-b818-002035559151 |

表 30. *cesProperty*

| 属性 | cesProperty |
|---|---|
| Active Directory LDAP 显示名称 | ibm-cesProperty |
| Active Directory 公共名（cn） | ibm-cesProperty |
| 描述 | 此属性的值可用来提供特定于应用程序的首选项配置参数。例如，值可以包含 XML 格式化数据。此属性的所有值在 cesPropertyType 属性值中都必须是同类。 |
| 语法 | 大小写精确的字符串 |
| 最大长度 | 32700 |
| 多值 | 多值 |
| OID（对象标识） | 1.3.18.0.2.4.307 |
| GUID（全局唯一标识） | b3afd2d5-5c5b-11d3-b818-002035559151 |

表 31. *cesPropertyType*

| 属性 | cesPropertyType |
|---|---|
| Active Directory LDAP 显示名称 | ibm-cesPropertyType |
| Active Directory 公共名（cn） | ibm-cesPropertyType |
| 描述 | 此属性的值可用来描述 cesProperty 属性的所有值的语法、语义或其他特征。例如，值“XML”可用来指示 cesProperty 属性的所有值都作为 XML 语法编码。 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 128 |
| 多值 | 多值 |
| OID（对象标识） | 1.3.18.0.2.4.308 |
| GUID（全局唯一标识） | b3afd2d6-5c5b-11d3-b818-002035559151 |

表 32. *cisProperty*

| 属性 | cisProperty |
|---|---|
| Active Directory LDAP 显示名称 | ibm-cisProperty |
| Active Directory 公共名（cn） | ibm-cisProperty |

*表 32. cisProperty (续)*

| 属性 | cisProperty |
| --- | --- |
| 描述 | 此属性的值可用来提供特定于应用程序的首选项配置参数。例如，值可以包含 INI 文件。此属性的所有值在其 cisPropertyType 属性值中都必须是同类。 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 32700 |
| 多值 | 多值 |
| OID（对象标识） | 1.3.18.0.2.4.309 |
| GUID（全局唯一标识） | b3afd2e0-5c5b-11d3-b818-002035559151 |

*表 33. cisPropertyType*

| 属性 | cisPropertyType |
| --- | --- |
| Active Directory LDAP 显示名称 | ibm-cisPropertyType |
| Active Directory 公共名（cn） | ibm-cisPropertyType |
| 描述 | 此属性的值可用来描述 cisProperty 属性的所有值的语法、语义或其他特征。例如，值“INI File”可用来指示 cisProperty 属性的所有值都是 INI 文件。 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 128 |
| 多值 | 多值 |
| OID（对象标识） | 1.3.18.0.2.4.310 |
| GUID（全局唯一标识） | b3afd2e1-5c5b-11d3-b818-002035559151 |

*表 34. binProperty*

| 属性 | binProperty |
| --- | --- |
| Active Directory LDAP 显示名称 | ibm-binProperty |
| Active Directory 公共名（cn） | ibm-binProperty |
| 描述 | 此属性的值可用来提供特定于应用程序的首选项配置参数。例如，值可以包含一组二进制编码的 Lotus® 123 属性。此属性的所有值在其 binPropertyType 属性值中都必须是同类。 |
| 语法 | 二进制 |
| 最大长度 | 250000 |
| 多值 | 多值 |
| OID（对象标识） | 1.3.18.0.2.4.305 |
| GUID（全局唯一标识） | b3afd2ba-5c5b-11d3-b818-002035559151 |

*表 35. binPropertyType*

| 属性 | binPropertyType |
| --- | --- |
| Active Directory LDAP 显示名称 | ibm-binPropertyType |
| Active Directory 公共名（cn） | ibm-binPropertyType |

表 35. *binPropertyType (续)*

| 属性 | **binPropertyType** |
|---|---|
| 描述 | 此属性的值可用来描述 binProperty 属性的所有值的语法、语义或其他特征。例如，值“Lotus 123”可用来指示 binProperty 属性的所有值都是二进制编码的 Lotus 123 属性。 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 128 |
| 多值 | 多值 |
| OID（对象标识） | 1.3.18.0.2.4.306 |
| GUID（全局唯一标识） | b3afd2bb-5c5b-11d3-b818-002035559151 |

表 36. *PropertyType*

| 属性 | **PropertyType** |
|---|---|
| Active Directory LDAP 显示名称 | ibm-propertyType |
| Active Directory 公共名（cn） | ibm-propertyType |
| 描述 | 此属性的值描述 eProperty 对象的语义特征 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 128 |
| 多值 | 多值 |
| OID（对象标识） | 1.3.18.0.2.4.320 |
| GUID（全局唯一标识） | b3afd4ed-5c5b-11d3-b818-002035559151 |

表 37. *settingID*

| 属性 | **settingID** |
|---|---|
| Active Directory LDAP 显示名称 | 不适用 |
| Active Directory 公共名（cn） | 不适用 |
| 描述 | 可用来标识 cimSetting 派生的对象条目（如 eProperty）的命名属性 |
| 语法 | 忽略大小写的字符串 |
| 最大长度 | 256 |
| 多值 | 单值 |
| OID（对象标识） | 1.3.18.0.2.4.325 |
| GUID（全局唯一标识） | b3afd596-5c5b-11d3-b818-002035559151 |

## 使用 DB2 对象类和属性来扩展 LDAP 目录模式

“LDAP 目录模式”定义了存储在 LDAP 目录条目中的信息的对象类和属性。对象类由一组必要的和可选的属性组成。LDAP 目录中的每一个条目都有一个与其相关联的对象类。

在 DB2 数据库管理器可以将信息存储在 LDAP 中之前，LDAP 服务器的“目录模式”必须包括 DB2 数据库系统使用的对象类和属性。向基本模式中添加新对象类和属性的过程称为*模式扩展*。

## 受支持的 LDAP 客户机和服务器配置

下表概述了受支持的 LDAP 客户机和服务器配置。

*表 38. 受支持的 LDAP 客户机和服务器配置*

| LDAP 客户机 | LDAP 服务器 | | |
|---|---|---|---|
| | IBM Tivoli Directory Server [1] | Microsoft Active Directory 服务器[2] | Sun One LDAP 服务器 |
| IBM LDAP 客户机[3] | 是否受支持 | 是否受支持 | 是否受支持 |
| Microsoft LDAP/ADSI 客户机[4] | 是否受支持 | 是否受支持 | 是否受支持 |

- [1] IBM Tivoli Directory Server 是一个 LDAP V3 服务器，并且可用于 Windows、AIX、Solaris、Linux 和 HP-UX。在 AIX 和 System i® 上，它与 OS/390® Security Server 一起作为基本操作系统的一部分交付。
- [2] Microsoft Active Directory 服务器是一个 LDAP V3 服务器，是作为操作系统的 Windows 2000 Server 和 Windows Server 2003 系列的部件提供的。
- [3] DB2 数据库系统支持安装在 AIX、Solaris、HP-UX 11.11、Windows 和 Linux 上的 IBM LDAP 客户机。
- [4] Microsoft LDAP 客户机是 Windows 操作系统附带包括的组件。

**注:** 在 Windows 操作系统上运行时，DB2 数据库管理器支持使用 IBM LDAP 客户机或 Microsoft LDAP 客户机。要显式地选择 IBM LDAP 客户机，可使用 db2set 命令将 **DB2LDAP_CLIENT_PROVIDER** 注册表变量设置为 IBM。Microsoft LDAP 客户机是 Windows 操作系统附带包括的。

## LDAP 支持和 DB2 Connect

如果 DB2 Connect 网关上提供了 LDAP 支持，且在网关数据库目录中找不到该数据库，那么 DB2 数据库管理器将在 LDAP 中查找数据库位置并尝试保留找到的信息。

### 在 LDAP 中注册主机数据库

在 LDAP 中注册主机数据库时，可以采用两种配置: 一种是直接连接至主机数据库，另一种是通过网关连接至主机数据库。

如果是直接连接至主机数据库，那么在 LDAP 中注册主机服务器，然后在 LDAP 中编目主机数据库，并指定主机服务器的节点名。如果是通过网关连接至主机数据库，那么在 LDAP 中注册网关服务器，然后在 LDAP 中编目主机数据库，并指定网关服务器的节点名。

如果 DB2 Connect 网关上提供了 LDAP 支持，且在网关数据库目录中找不到该数据库，那么 DB2 数据库系统将查找 LDAP 并尝试保留找到的信息。

以下示例显示了上述两种情况，请考虑以下情况: 假定有一个称为 NIAGARA_FALLS 的主机数据库。它可以接受使用 TCP/IP 协议的入局连接。如果客户机因没有 DB2 Connect 而不能直接连接至主机，那么它将使用称为 goto@niagara 的网关进行连接。

需要完成下列步骤:

1. 在 LDAP 中为 TCP/IP 连接注册主机数据库服务器。服务器的 TCP/IP 主机名是“myhost”，端口号是“446”。**NODETYPE** 子句将设置为 DCS 以指示这是一个主机数据库服务器。

   ```
   db2 register ldap as nftcpip tcpip hostname myhost svcename 446
      remote mvssys instance mvsinst nodetype dcs
   ```

2. 在 LDAP 中为 TCP/IP 连接注册 DB2 Connect 网关服务器。网关服务器的 TCP/IP 主机名是“niagara”，端口号是“50000”。

   ```
   db2 register ldap as whasf tcpip hostname niagara svcename 50000
      remote niagara instance goto nodetype server
   ```

3. 使用 TCP/IP 连接在 LDAP 中编目主机数据库。主机数据库名称是“NIAGARA_FALLS”，数据库别名是“nftcpip”。**GWNODE** 子句用来指定 DB2 Connect 网关服务器的节点名。

   ```
   db2 catalog ldap database NIAGARA_FALLS as nftcpip at node nftcpip
   gwnode whasf authentication server
   ```

在完成上面显示的注册和编目之后，如果要使用 TCPIP 连接主机，那么连接“nftcpip”。如果客户机工作站上没有安装 DB2 Connect，那么将通过网关并使用 TCPIP 进行连接。在网关中，它使用 TCPIP 连接至主机。

通常，可在 LDAP 中手动配置主机数据库信息，以便每台客户机不必在自己的机器上以本地方式手动编目该数据库和节点。该过程如下:

1. 在 LDAP 中注册主机数据库服务器。在 REGISTER 命令中，必须分别使用 **REMOTE**、**INSTANCE** 和 **NODETYPE** 子句指定远程计算机名称、实例名和主机数据库服务器的节点类型。**REMOTE** 子句可以设置为主机名或主机服务器的 LU 名。**INSTANCE** 子句可以设置为长度不超过 8 个字符的任何字符串。（例如，可以将实例名设置为“DB2”。）必须将 **NODETYPE** 子句设置为 DCS 以指示这是一个主机数据库服务器。
2. 使用 CATALOG LDAP DATABASE 命令在 LDAP 中注册主机数据库。可使用 **PARMS** 子句指定任何其他 DRDA® 参数。应将数据库认证类型设置为 SERVER。

## 扩展 IBM Tivoli Directory Server 的目录模式

如果使用的是 IBM Tivoli Directory Server，那么 DB2 V8.2 之前的数据库需要的所有对象类和属性均包括在基本模式中。

运行以下命令来扩展具有在版本 8.2 和更高版本中引入的新 DB2 数据库属性的基本模式:

```
ldapmodify -c -h machine_name:389 -D dn -w password -f altgwnode.ldif
```

以下是 `altgwnode.ldif` 文件的内容:

```
dn:                cn=schema
      changetype:    modify
      add:               attributetypes
      attributetypes:    (
        1.3.18.0.2.4.3092
        NAME 'db2altgwPtr'
 DESC 'DN pointer to DB2 alternate gateway (node) object'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
-
      add:               ibmattributetypes
      ibmattributetypes: (
        1.3.18.0.2.4.3092
        DBNAME ('db2altgwPtr' 'db2altgwPtr')
        ACCESS-CLASS NORMAL
        LENGTH 1000)


      dn:            cn=schema
      changetype:    modify
      add:               attributetypes
      attributetypes:    (
        1.3.18.0.2.4.3093
        NAME 'db2altnodePtr'
 DESC 'DN pointer to DB2 alternate node object'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.12)
-
      add:               ibmattributetypes
      ibmattributetypes: (
        1.3.18.0.2.4.3093
        DBNAME ('db2altnodePtr' 'db2altnodePtr')
        ACCESS-CLASS NORMAL
        LENGTH 1000)


      dn:            cn=schema
      changetype:    modify
      replace:       objectclasses
      objectclasses: (
 1.3.18.0.2.6.117              NAME 'DB2Database'
        DESC 'DB2 database'
        SUP cimSetting
        MUST ( db2databaseName $ db2nodePtr )
 MAY ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr
    $ db2ARLibrary $ db2authenticationLocation $ db2databaseAlias
    $ db2databaseRelease $ db2gwPtr $ DCEPrincipalName ) )
```

可以在如下 URL 找到 altgwnode.ldif 和 altgwnode.readme 文件: ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap

在添加 DB2 模式定义之后，必须重新启动“目录服务器”以激活所有更改。

## Netscape LDAP 目录支持和属性定义

受支持的 Netscape LDAP 服务器级别是 V4.12 或更高版本。

在 Netscape LDAP 服务器版本 4.12 或更高版本中，“Netscape 目录服务器”允许应用程序通过将属性和对象类定义添加至 slapd.user_oc.conf 和 slapd.user_at.conf 这两个文件来扩展模式。这两个文件位于 <Netscape_install path>\slapd-<machine_name>\config 目录中。

**注**: 如果正在使用 Sun One Directory Server 5.0，请参阅关于扩展 Sun One Directory Server 的目录模式的主题。

必须将 DB2 属性添加至 slapd.user_at.conf，如下所示:

```
##############################################################################
#
      # IBM DB2 Database
 # Attribute Definitions
#
 # bin —> binary
 # ces —> case exact string
 # cis —> case insensitive string
 # dn —> distinguished name
#
##############################################################################

 attribute binProperty                    1.3.18.0.2.4.305     bin
 attribute binPropertyType                1.3.18.0.2.4.306     cis
 attribute cesProperty                    1.3.18.0.2.4.307     ces
 attribute cesPropertyType                1.3.18.0.2.4.308     cis
 attribute cisProperty                    1.3.18.0.2.4.309     cis
 attribute cisPropertyType                1.3.18.0.2.4.310     cis
 attribute propertyType                   1.3.18.0.2.4.320     cis
 attribute systemName                     1.3.18.0.2.4.329     cis
 attribute db2nodeName                    1.3.18.0.2.4.419     cis
 attribute db2nodeAlias                   1.3.18.0.2.4.420     cis
 attribute db2instanceName                1.3.18.0.2.4.428     cis
 attribute db2Type                        1.3.18.0.2.4.418     cis
 attribute db2databaseName                1.3.18.0.2.4.421     cis
 attribute db2databaseAlias               1.3.18.0.2.4.422     cis
 attribute db2nodePtr                     1.3.18.0.2.4.423     dn
 attribute db2gwPtr                       1.3.18.0.2.4.424     dn
 attribute db2additionalParameters        1.3.18.0.2.4.426     cis
 attribute db2ARLibrary                   1.3.18.0.2.4.427     cis
 attribute db2authenticationLocation      1.3.18.0.2.4.425     cis
 attribute db2databaseRelease             1.3.18.0.2.4.429     cis
 attribute DCEPrincipalName               1.3.18.0.2.4.443     cis
```

必须将 DB2 对象类添加至 slapd.user_oc.conf 文件，如下所示:

```
##############################################################################
#
      # IBM DB2 Database
# Object Class Definitions
#
##############################################################################

objectclass eProperty
                oid 1.3.18.0.2.6.90
                requires
                                objectClass
                allows
                                cn,
                                propertyType,
                                binProperty,
                                binPropertyType,
                                cesProperty,
                                cesPropertyType,
                                cisProperty,
                cisPropertyType
objectclass eApplicationSystem
                oid 1.3.18.0.2.6.84
                requires
                                objectClass,
                                systemName
```

```
objectclass DB2Node
                oid 1.3.18.0.2.6.116
                requires
                                objectClass,
                db2nodeName                allows
                                db2nodeAlias,
                                host,
                                db2instanceName,
                                db2Type,
                                description,
                                protocolInformation

objectclass DB2Database
                oid 1.3.18.0.2.6.117
                requires
                                objectClass,
                                db2databaseName,
                db2nodePtr                 allows
                                db2databaseAlias,
                                description,
                                db2gwPtr,
                                db2additionalParameters,
                                db2authenticationLocation,
                                DCEPrincipalName,
                                db2databaseRelease,
                db2ARLibrary
```

在添加 DB2 模式定义之后，必须重新启动“目录服务器”以激活所有更改。

## 展开 Sun One Directory Server 的目录模式

Sun One Directory Server 也称为 Netscape 或 iPlanet Directory Server。

要使 Sun One Directory Server 在您的环境中工作，应将 60ibmdb2.ldif 文件添加到以下目录中:

在 Windows 上，如果将 iPlanet 安装在 C:\iPlanet\Servers 中，那么将上述文件添加到 .\sldap-<machine_name>\config\schema。

在 UNIX 上，如果将 iPlanet 安装在 /usr/iplanet/servers 中，那么将上述文件添加到 ./slapd-<machine_name>/config/schema。

以下是该文件的内容:

```
#########################################################################
      # IBM DB2 Database
#########################################################################
      dn:            cn=schema
#########################################################################
      # Attribute Definitions (Before V8.2)
#########################################################################
      attributetypes: ( 1.3.18.0.2.4.305  NAME 'binProperty'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.5              X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.306  NAME 'binPropertyType'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.307  NAME 'cesProperty'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.308  NAME 'cesPropertyType'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.309  NAME 'cisProperty'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.310  NAME 'cisPropertyType'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
```

```
      attributetypes: ( 1.3.18.0.2.4.320  NAME 'propertyType'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.329  NAME 'systemName'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.419  NAME 'db2nodeName'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.420  NAME 'db2nodeAlias'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.428  NAME 'db2instanceName'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.418  NAME 'db2Type'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.421  NAME 'db2databaseName'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.422  NAME 'db2databaseAlias'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15             X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.426  NAME 'db2additionalParameters'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.427  NAME 'db2ARLibrary'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.425  NAME 'db2authenticationLocation'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.429  NAME 'db2databaseRelease'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.443  NAME 'DCEPrincipalName'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.423  NAME 'db2nodePtr'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.424  NAME 'db2gwPtr'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
#############################################################################
# Attribute Definitions (V8.2 and later)
#############################################################################
      attributetypes: ( 1.3.18.0.2.4.3092 NAME 'db2altgwPtr'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12             X-ORIGIN 'IBM DB2' )
      attributetypes: ( 1.3.18.0.2.4.3093 NAME 'db2altnodePtr'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12             X-ORIGIN 'IBM DB2' )
#############################################################################
# Object Class Definitions
      # DB2Database for V8.2 has the above two new optional attributes.
#############################################################################
objectClasses: ( 1.3.18.0.2.6.90  NAME 'eProperty'
   SUP top STRUCTURAL MAY ( cn $ propertyType $ binProperty
   $ binPropertyType $ cesProperty $ cesPropertyType $ cisProperty
   $ cisPropertyType ) X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.84  NAME 'eApplicationSystem'
   SUP top STRUCTURAL MUST systemName
   X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.116 NAME 'DB2Node'
   SUP top STRUCTURAL MUST db2nodeName MAY ( db2instanceName $ db2nodeAlias
   $ db2Type $ description $ host $ protocolInformation )
   X-ORIGIN 'IBM DB2' )
objectClasses: ( 1.3.18.0.2.6.117 NAME 'DB2Database'
   SUP top STRUCTURAL MUST (db2databaseName $ db2nodePtr ) MAY
   ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr $ db2ARLibrary
   $ db2authenticationLocation $ db2databaseAlias $ db2databaseRelease
   $ db2gwPtr $ DCEPrincipalName $ description )
   X-ORIGIN 'IBM DB2' )
```

60ibmdb2.ldif 和 60ibmdb2.readme 文件位于 URL: ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap

在添加 DB2 模式定义之后，必须重新启动“目录服务器”以激活所有更改。

# Windows Active Directory

DB2 数据库服务器作为 ibm_db2Node 对象发布在 Active Directory 中。ibm_db2Node 对象类是 ServiceConnectionPoint(SCP) 对象类的子类。

每个 ibm_db2Node 对象都包含协议配置信息以允许客户机应用程序连接至 DB2 数据库服务器。创建新数据库时，在 Active Directory 中，将该数据库作为 ibm_db2Database 对象发布在 ibm_db2Node 对象下面。

当连接至远程数据库时，DB2 客户机通过 LDAP 接口查询 Active Directory，以查找 ibm_db2Database 对象。用于连接数据库服务器的协议通信（绑定信息）是从 ibm_db2Node 对象获取的，ibm_db2Database 对象在该对象下面创建。

可在域控制器上使用 *Active Directory 用户和计算机*“管理控制台”（MMC）来查看或修改 ibm_db2Node 和 ibm_db2Database 对象的属性页。要设置属性页，运行 regsrv32 命令以按如下所示注册 DB2 对象的属性页:

```
regsvr32 %DB2PATH%\bin\db2ads.dll
```

可在域控制器上使用 *Active Directory 用户和计算机*“管理控制台”（MMC）来查看对象。要获取此管理工具，依次单击“开始”—>“程序”—>“管理工具”—>“Active Directory 用户和计算机”。

**注:** 必须从“视图”菜单中选择*用户、组和计算机（作为容器）*来显示计算机对象下面的 DB2 数据库对象。

**注:** 如果 DB2 数据库系统未安装在域控制器上，仍可通过将 %DB2PATH%\bin 中的 db2ads.dll 文件以及 %DB2PATH%\msg\locale-name 中的资源 DLL db2adsr.dll 复制至域控制器上的本地目录来查看 DB2 数据库对象的属性页。（这两个复制文件所在的目录必须是可在 PATH 环境变量中找到的其中一个目录。）然后，从本地目录运行 regsvr32 命令来注册 DLL。

## 配置 DB2 数据库管理器以使用 Active Directory

为了访问 Microsoft Active Directory，确保符合下列条件:

1. 运行 DB2 数据库的计算机必须属于 Windows 2000 或 Windows Server 2003 域。
2. 已安装 Microsoft LDAP 客户机。Microsoft LDAP 客户机是 Windows 2000、Windows XP 和 Windows Server 2003 操作系统的一部分。
3. 启用 LDAP 支持。有关更多信息，请参阅《*DB2 服务器快速入门*》中的『扩展 Active Directory Schema for LDAP 目录服务（Windows）』。
4. 当运行 DB2 数据库系统时登录域用户帐户以便从 Active Directory 读取信息。

## Active Directory 的安全性注意事项

DB2 数据库和节点对象是在将 DB2 服务器安装在 Active Directory 中的机器的计算机对象下创建的。要在 Active Directory 中注册数据库服务器或者对数据库进行编目，您需要具有充分的访问权才能创建或更新计算机对象下的对象。

在缺省情况下，计算机对象下的对象可由任何经认证的用户读取，并可以由管理员（属于“管理员”、“域管理员”和“企业管理员”组的用户）更新。要授予特定用户或组的访问权，可使用 *Active Directory 用户和计算机*“管理控制台”（MMC），如下所示:

1. 启动 *Active Directory 用户和计算机*管理工具

（“开始”—>“程序”—>“管理工具”—>“Active Directory 用户和计算机”）

2. 在*查看*下面，选择*高级功能*
3. 选择*计算机*容器
4. 右键单击表示安装有 DB2 的服务器的计算机对象，并选择*属性*
5. 选择*安全性*选项卡，然后向指定的用户或组添加必需的访问权

用户级的 DB2 注册表变量和 CLI 设置保存在“用户”对象下面的 DB2 属性对象中。要在用户级设置 DB2 注册表变量或 CLI 设置，用户需要具有足够的访问权才能在“用户”对象下面创建对象。

在缺省情况下，只有管理员才具有在“用户”对象下面创建对象的访问权。要授予用户在用户级设置 DB2 注册表变量或 CLI 设置的访问权，可使用 *Active Directory 用户和计算机*“管理控制台”（MMC），如下所示:

1. 启动 *Active Directory 用户和计算机*管理工具

（“开始”—>“程序”—>“管理工具”—>“Active Directory 用户和计算机”）

2. 在“用户”容器下面选择该用户对象
3. 右键单击该用户对象并选择*属性*
4. 选择*安全性*选项卡
5. 使用添加“按钮”将用户名添加至列表
6. 授予“写入”和“创建所有子对象”访问权
7. 使用“高级”设置，设置许可权以应用到“此对象和所有子对象”
8. 选中“允许父代的可继承许可权传播至此对象”复选框

## Active Directory 中的 DB2 对象

DB2 数据库管理器在 Active Directory 中的以下两个位置创建对象:

1. 在安装了 DB2 服务器的机器的计算机对象下面创建 DB2 数据库和节点对象。对于不属于 Windows 域的 DB2 服务器，在“系统”容器下面创建 DB2 数据库和节点对象。
2. 用户级的 DB2 注册表变量和 CLI 设置存储在“用户”对象下面的 DB2 属性对象中。这些对象包含该用户的特定信息。

## 扩展 Active Directory 的目录模式

在 DB2 数据库管理器可以将信息存储在 Active Directory 中之前，需要扩展目录模式以包括新的 DB2 数据库对象类和属性。向目录模式中添加新对象类和属性的过程称为*模式扩展*。

必须通过运行 DB2 模式安装程序 db2schex 来扩展 Active Directory 的模式。应该先运行此命令，然后安装 DB2 产品并创建数据库，否则您必须手动注册节点并对这些数据库进行编目。

db2schex 程序包括在产品 CD-ROM 中，位置如下: `x:\db2\windows\utilities\`，其中 x: 是 DVD 盘符。

要更新模式，您必须是“模式管理员”组的成员，或已被授予更新模式的权限。在属于 Windows 域的任何机器上运行以下命令:

```
runas /user:MyDomain\Administrator x:\db2\Windows\utilities\db2schex.exe
```

其中 x: 表示 DVD 盘符。

如果已运行较早版本的 DB2 数据库管理系统中的 db2schex 命令，那么当您在 DB2 UDB 版本 8.2 或更高版本中再次运行此命令时，就会将以下两个可选属性添加至 ibm-db2Database 类:

```
ibm-db2AltGwPtr
ibm-db2NodePtr
```

如果尚未在 Windows 上的较早版本的 DB2 数据库管理系统中运行 db2schex 命令，那么当您在 DB2 版本 9.5 或更高版本中运行此命令时，就会添加 DB2 数据库系统 LDAP 支持的所有类和属性。

此命令还有其他可选子句。有关更多信息，请参阅“db2schex - Active Directory 模式扩展命令”主题。

示例:

- 要安装 DB2 数据库模式:

  ```
  db2schex
  ```

- 要安装 DB2 数据库模式并指定绑定 DN 和密码:

  ```
  db2schex -b "cn=A Name,dc=toronto1,dc=ibm,dc=com"
           -w password
  ```

  或者,

  ```
  db2schex -b Administrator -w password
  ```

- 要卸载 DB2 数据库模式:

  ```
  db2schex -u
  ```

- 要卸载 DB2 数据库模式并忽略错误:

  ```
  db2schex -u -k
  ```

## 完成 DB2 安装之后启用 LDAP 支持

必须在完成 DB2 产品安装后启用 LDAP，然后才能使用 LDAP。

要启用 LDAP 支持:

1. 在任何属于 Windows 域的机器上，执行以下步骤:
   a. 如果未这样做就安装 DB2 产品，那么您必须扩展目录模式（如果要使用 Microsoft Active Directory）。有关更多信息，请参阅“扩展 Active Directory 的目录模式”主题。
   b. 通过运行 DB2 安装程序并从定制安装中选择“LDAP 目录开发”支持来安装 LDAP 支持二进制文件。 安装程序会自动将 DB2 注册表变量 **DB2_ENABLE_LDAP** 设置为 YES，这是启用 LDAP 支持所必需的设置。
   c. 可选: 要使用 IBM LDAP 客户机取代 Microsoft LDAP 客户机，请将 **DB2LDAP_CLIENT_PROVIDER** 注册表变量设置为 IBM。
2. 在每个 LDAP 客户机上，执行以下步骤:
   a. 通过运行以下命令来指定 LDAP 服务器的 TCP/IP 主机名（还可以选择指定端口号）: db2set DB2LDAPHOST=*base_domain_name*[:port_number]，其中 *base_domain_name* 是 TCP/IP 主机名，而 *[:port_number]* 是端口号。如果未指定端口号，那么将使用缺省 LDAP 端口号 389。对于已启用 SSL 的 LDAP 服

务器，运行以下命令：db2set DB2LDAPHOST=*base_domain_name*:SSL:636，其中 *base_domain_name* 是 TCP/IP 主机名。

DB2 对象位于 LDAP 基本专有名称（baseDN）中。可以通过运行以下命令来配置每台机器上的基本专有名称（baseDN）：

```
db2set DB2LDAP_BASEDN=baseDN
```

其中 *baseDN* 是 LDAP 服务器上定义的 LDAP 后缀的名称。

b. 可选：要使用 LDAP 来存储特定于 DB2 用户的信息，请输入此 LDAP 用户的专有名称（DN）和密码。

3. 如果您在安装 DB2 产品之后扩展了目录模式，请执行下列步骤：
   a. 通过运行以下命令在 LDAP 中注册 DB2 服务器的当前实例：

      ```
      db2 register ldap as node-name protocol tcpip
      ```

   b. 通过运行以下命令在 LDAP 中注册特定数据库：

      ```
      db2 catalog ldap database dbname as alias_dbname
      ```

现在，您就可以注册 LDAP 条目。

**注**：将 **DB2LDAPHOST** 设置为已启用 SSL 的 LDAP 服务器只允许从使用 SSL 的 LDAP 服务器中检索目录信息。要确保使用 SSL 加密在客户机与服务器之间交换的数据，请参阅在 DB2 客户机中配置安全套接字层 (SSL) 支持以了解更多信息。

# 注册 LDAP 条目

## 安装之后注册 DB2 服务器

必须在 LDAP 中注册每个 DB2 服务器实例，以便发布客户机应用程序用来连接 DB2 服务器实例的协议配置信息。

当注册一个数据库服务器实例时，需要指定一个*节点名*。该节点名由客户机应用程序在连接服务器时使用。可使用 CATALOG LDAP NODE 命令编目 LDAP 节点的另一别名。

**注**：如果是在 Windows 域环境中工作，那么安装期间，将在 Active Directory 中用下列信息自动注册 DB2 服务器实例：

```
nodename: TCP/IP hostname
protocol type: TCP/IP
```

如果 TCP/IP 主机名长于 8 个字符，那么它将被截断为 8 个字符。

REGISTER 命令如下所示：

```
db2 register db2 server in ldap
   as <ldap_node_name>
   protocol tcpip
```

protocol 子句指定与此数据库服务器连接时要使用的通信协议。

为包含多台物理机器的 DB2 企业服务器版创建实例时，必须对每台机器调用一次 REGISTER 命令。使用 rah 命令在所有机器上发出 REGISTER 命令。

**注:** 每台机器不能使用相同的 ldap_node_name，因为在 LDAP 中该名称必须唯一。以每台机器的主机名替换 REGISTER 命令中的 ldap_node_name。例如:

```
rah ">DB2 REGISTER DB2 SERVER IN LDAP AS <> PROTOCOL TCPIP"
```

“<>”将替换为运行 rah 命令的每台机器的主机名。如果出现有多个 DB2 企业服务器版实例这种极少见的情况，那么可将实例与主机索引的组合用作 rah 命令中的节点名。

可对远程 DB2 服务器发出 REGISTER 命令。为此，当注册远程服务器时，必须指定远程计算机名称、实例名和协议配置参数。可以按如下所示使用该命令:

```
db2 register db2 server in ldap
   as <ldap_node_name>
         protocol tcpip
   hostname <host_name>
   svcename <tcpip_service_name>
   remote <remote_computer_name>
   instance <instance_name>
```

以下是计算机名称的约定:

- 如果配置了 TCP/IP，那么计算机名称必须与 TCP/IP 主机名相同。

当在一个高可用性或故障转移的环境中运行，并使用 TCP/IP 作为通信协议时，必须使用集群 IP 地址。使用集群 IP 地址允许客户机连接到任何一台机器上的服务器，而不必为每台机器编目独立的 TCP/IP 节点。使用 hostname 子句指定集群 IP 地址，如下所示:

```
db2 register db2 server in ldap
   as <ldap_node_name>
         protocol tcpip
         hostname n.nn.nn.nn
```

其中 n.nn.nn.nn 是集群 IP 地址。

要从客户机应用程序注册 LDAP 中的 DB2 服务器，请调用 db2LdapRegister API。

## 编目节点别名以进行连接 (ATTACH)

当在 LDAP 中注册服务器时，必须指定 DB2 服务器的节点名。应用程序使用该节点名连接数据库服务器。

如果需要另一个节点名（例如，将节点名硬编码到应用程序中时），那么可使用 CATALOG LDAP NODE 命令进行更改，例如:

```
db2 catalog ldap node <ldap_node_name>
   as <new_alias_name>
```

要取消编目 LDAP 节点，可使用 UNCATALOG LDAP NODE 命令，例如:

```
db2 uncatalog ldap node <ldap_node_name>
```

## 在 LDAP 目录中注册数据库

在实例中创建数据库期间，会在 LDAP 中自动注册数据库。注册允许远程客户机与数据库连接，而不必在客户机上编目该数据库和节点。当客户机试图连接数据库时，如果本地机器上的数据库目录中不存在该数据库，那么搜索 LDAP 目录。

如果 LDAP 目录中已存在该名称，仍然会在本地机器上创建该数据库，但是会返回一个警告消息，说明在 LDAP 目录中发生名称冲突。因此，可在 LDAP 目录中手动编目数

据库。用户可使用 CATALOG LDAP DATABASE 命令在 LDAP 中注册远程服务器上的数据库。当注册远程数据库时，指定表示远程数据库服务器的 LDAP 节点的名称。在注册数据库之前，必须使用 REGISTER DB2 SERVER IN LDAP 命令在 LDAP 中注册远程数据库服务器。要在 LDAP 中手动注册数据库，可使用 CATALOG LDAP DATABASE 命令:

```
db2 catalog ldap database <dbname>
at node <node_name>
      with "My LDAP database"
```

要从客户机应用程序注册 LDAP 中的数据库，请调用 db2LdapCatalogDatabase API。

# 注销 LDAP 条目

## 注销 DB2 服务器

从 LDAP 中注销一个实例，同时也除去了所有引用该实例的节点、别名、对象和数据库对象。

要在本地机器或远程机器上注销 DB2 服务器，要求为服务器指定 LDAP 节点名:

```
db2 deregister db2 server in ldap
   node <node_name>
```

要从客户机应用程序注销 LDAP 中的 DB2 服务器，请调用 db2LdapDeregister API。

当注销了 DB2 服务器时，引用 DB2 服务器的同一个实例的任何 LDAP 节点条目和 LDAP 数据库条目也将被取消编目。

## 从 LDAP 目录中注销数据库

当删除数据库或者从 LDAP 中注销拥有实例时，就会从 LDAP 中自动注销数据库。

可以使用以下命令从 LDAP 中手动注销数据库:

```
db2 uncatalog ldap database <dbname>
```

要从客户机应用程序注销 LDAP 中的数据库，请调用 db2LdapUncatalogDatabase API。

# 配置 LDAP 用户

## 创建 LDAP 用户

DB2 数据库系统支持在用户级设置 DB2 注册表变量和 CLI 配置。（在 Linux 和 UNIX 平台上，此功能不可用。）用户级支持在多用户环境中提供了用户特定设置。Windows Terminal Server 便是一个示例，每个登录用户都可以定制他/她自己的环境，而不会干扰系统环境或另一用户的环境。

在使用 IBM Tivoli 目录时，必须先定义 LDAP 用户，然后才可以在 LDAP 中存储用户级信息。通过创建 LDIF 文件来创建一个 LDAP 用户，以包含用户对象的所有属性，然后运行 LDIF IMPORT 实用程序将该对象导入到 LDAP 目录中。用于 IBM Tivoli Directory Server 的 LDIF 实用程序是 LDIF2DB。

包含人员对象属性的 LDIF 文件类似于以下内容:

```
File name: newuser.ldif

dn: cn=Mary Burnnet, ou=DB2 Development, ou=Toronto, o=ibm, c=ca
objectclass: ePerson
cn: Mary Burnnet
sn: Burnnet
uid: mburnnet
userPassword: password
telephonenumber: 1-416-123-4567
facsimiletelephonenumber: 1-416-123-4568
title: Software Developer
```

以下是 LDIF 命令的示例，该命令将使用 IBM LDIF IMPORT 实用程序来导入 LDIF 文件:

```
LDIF2DB -i newuser.ldif
```

**注:**

1. 必须从 LDAP 服务器运行 LDIF2DB 命令。
2. 必须将必需的访问权（ACL）授予 LDAP 用户对象，以使 LDAP 用户可以添加、删除、读取和写入他自己的对象。要授予用户对象的 ACL，使用“LDAP 目录服务器 Web 管理”工具。

## 为 DB2 应用程序配置 LDAP 用户

使用 Microsoft LDAP 客户机时，LDAP 用户与操作系统用户帐户相同。但是，使用 IBM LDAP 客户机时，在使用 DB2 数据库管理器之前，必须配置当前登录用户的 LDAP 用户专有名称（DN）和密码。

要配置 LDAP 用户的专有名称（DN）和密码，使用 db2ldcfg 实用程序:

```
db2ldcfg -u <userDN> -w <password> --> set the user's DN and password
         -r                        --> clear the user's DN and password
```

例如:

```
db2ldcfg -u "cn=Mary Burnnet,ou=DB2 Development,ou=Toronto,o=ibm,c=ca"
         -w password
```

## 在 LDAP 环境中设置用户级的 DB2 注册表变量

在 LDAP 环境中，可在用户级设置 DB2 概要文件注册表变量，这样可允许用户定制自己的 DB2 环境。

要在用户级设置 DB2 概要文件注册表变量，可使用 -ul 选项:

```
db2set -ul <variable>=<value>
```

**注:** 这在 AIX 或 Solaris 操作系统上不受支持。

DB2 有高速缓存机制。将用户级 DB2 概要文件注册表变量高速缓存到本地机器上。如果指定了 -ul 参数，DB2 将始终从高速缓存中读取 DB2 注册表变量。当发生下列情况时，会刷新高速缓存:

- 更新或复位用户级 DB2 注册表变量。
- 在用户级刷新 LDAP 概要文件变量的命令是:

```
db2set -ur
```

## 禁用 LDAP 支持

要禁用 LDAP 支持，使用下列过程:

1. 对每个 DB2 服务器实例，注销 LDAP 中的 DB2 服务器:

```
db2 deregister db2 server in ldap node <nodename>
```

2. 将 DB2 概要文件注册表变量 DB2_ENABLE_LDAP 设置为“NO”。

## 更新 DB2 服务器的协议信息

LDAP 中的 DB2 服务器信息必须保持为最新信息。例如，更改协议配置参数或服务器网络地址时要求更新 LDAP。

要更新本地机器上的 LDAP 中的 DB2 服务器，使用以下命令:

```
db2 update ldap ...
```

可更新的协议配置参数的示例包括: TCP/IP 主机名以及服务名称或端口号参数。

要更新远程 DB2 服务器协议配置参数，请使用带有 node 子句的 UPDATE LDAP 命令:

```
db2 update ldap
   node <node_name>
   hostname <host_name>
   svcename <tcpip_service_name>
```

## 将 LDAP 客户机重新路由至另一台服务器

正如系统发生故障时可以重新路由客户机一样，使用 LDAP 时，您也具有相同的能力。

DB2_ENABLE_LDAP 注册表变量必须设置为“YES”。

在 LDAP 环境中，所有数据库和节点目录信息保留在 LDAP 服务器中。客户机从 LDAP 目录中检索信息。如果 DB2LDAPCACHE 注册表变量设置为“Yes”，那么在其本地数据库和节点目录中更新此信息。

使用 UPDATE ALTERNATE SERVER FOR LDAP DATABASE 命令为 LDAP 中代表 DB2 数据库的数据库定义备用服务器。或者，可以从客户机应用程序调用 db2LdapUpdateAlternateServerForDB API 来更新 LDAP 中数据库的备用服务器。

建立之后，连接时将此备用服务器信息返回至客户机。

**注**: 强烈建议保持 LDAP 服务器中存储的替代服务器信息与数据库服务器实例上存储的替代服务器信息同步。在数据库服务器实例上发出 UPDATE ALTERNATE SERVER FOR DATABASE 命令（注意，不是“FOR LDAP DATABASE”）有助于确保数据库服务器实例与 LDAP 服务器同步。

在服务器实例上输入 UPDATE ALTERNATE SERVER FOR DATABASE 命令时，如果在该服务器上已启用了 LDAP 支持（DB2_ENABLE_LDAP=Yes），并且已对 LDAP 用户标识和密码进行高速缓存（先前已运行了 db2ldcfg），那么将在 LDAP 服务器上自动地或隐式地更新数据库的替代服务器。这与显式地输入 UPDATE ALTERNATE SERVER FOR LDAP DATABASE 效果相同。

如果从除数据库服务器实例以外的实例发出 UPDATE ALTERNATE SERVER FOR LDAP DATABASE 命令，那么确保在数据库服务器实例上使用 UPDATE ALTERNATE SERVER FOR DATABASE 命令来配置完全相同的替代服务器信息。在客户机最初连接到数据库服务器实例后，从数据库服务器实例返回的替代服务器信息将优先于 LDAP 服务器中配置的信息。如果在数据库服务器实例上未配置替代服务器信息，在初始连接后将认为客户机重新路由功能处于禁用状态。

## 在 LDAP 环境中连接至远程服务器

在 LDAP 环境中，可在 ATTACH 命令中使用 LDAP 节点名连接至远程数据库服务器: `db2 attach to <ldap_node_name>`。

当客户机应用程序首次连接节点或连接至数据库时，由于该节点不在本地节点目录中，数据库管理器会搜索 LDAP 目录以查找目标节点条目。如果在 LDAP 目录中找到该条目，就会检索远程服务器的协议信息。如果连接的是数据库，且在 LDAP 目录中找到该条目，那么还检索数据库信息。使用该信息，数据库管理器自动在本地机器上对数据库条目和节点条目编目。客户机应用程序下次连接相同的节点或数据库时，可使用本地数据库目录中的信息，而不必搜索 LDAP 目录。

具体地说，由于存在高速缓存机制，因此客户机仅搜索 LDAP 服务器一次。检索到信息之后，就根据数据库管理器配置参数 *dir_cache* 和注册表变量 DB2LDAPCACHE 的值，在本地机器上存储或高速缓存此信息。

- 如果 DB2LDAPCACHE=NO 且 dir_cache=NO，那么始终从 LDAP 中读取信息。
- 如果 DB2LDAPCACHE=NO 但 dir_cache=YES，那么从 LDAP 中读取一次信息，并将其插入到 DB2® 高速缓存中。
- 如果设置 DB2LDAPCACHE=YES 或未设置 DB2LDAPCACHE 的值，那么从 LDAP 服务器中读取一次信息，并将其高速缓存至本地数据库、节点和 DCS 目录中。

**注**: LDAP 信息的高速缓存不适用于用户级 CLI 或 DB2 概要文件注册表变量。

## 刷新本地数据库和节点目录中的 LDAP 条目

DB2 数据库系统提供了一种高速缓存机制来减少客户机搜索 LDAP 服务器的次数。

检索到信息之后，就根据数据库管理器配置参数 dir_cache 和注册表变量 DB2LDAPCACHE 的值，在本地机器上存储或高速缓存此信息。

- 如果 DB2LDAPCACHE=NO 且 dir_cache=NO，那么始终从 LDAP 中读取信息。
- 如果 DB2LDAPCACHE=NO 但 dir_cache=YES，那么从 LDAP 中读取一次信息，并将其插入到 DB2 高速缓存中。
- 如果设置 DB2LDAPCACHE=YES 或未设置 DB2LDAPCACHE 的值，那么从 LDAP 服务器中读取一次信息，并将其高速缓存至本地数据库、节点和 DCS 目录中。

**注**: LDAP 信息的高速缓存不适用于用户级 CLI 或 DB2 概要文件注册表变量。由于 LDAP 中的信息可能会更改，因此有必要刷新高速缓存在本地数据库和节点目录中的 LDAP 条目。可以采用一些方法来执行此操作。

要刷新从 LDAP 中检索到的所有本地数据库和节点条目，请使用以下命令:

```
db2 refresh ldap immediate
```

同样，可以使用以下命令来刷新现有本地数据库和节点条目并在 LDAP 中添加新条目:

```
db2 refresh ldap immediate all
```

指定 IMMEDIATE ALL 选项可将随 LDAP 服务器包含的所有 NODE 和 DB 条目添加到本地目录中。

此外，要强制 DB2 在下一个数据库连接或实例连接时刷新引用 LDAP 资源的数据库条目，请使用以下命令:

```
db2 refresh ldap database directory
```

同样，要强制 DB2 在下一个数据库连接或实例连接时刷新引用 LDAP 资源的节点条目，请使用以下命令:

```
db2 refresh ldap node directory
```

在刷新过程中，会除去本地数据库和节点目录中保存的所有 LDAP 条目。下次应用程序访问该数据库或节点时，它将直接从 LDAP 中读取该信息并在本地数据库或节点目录中生成新条目。

要确保能够及时进行刷新，可以采用以下几种方法:

- 安排定期运行刷新。
- 在系统引导期间运行 REFRESH 命令。
- 使用提供的管理包在所有客户机上调用 REFRESH 命令。
- 将 DB2LDAPCACHE 设置为“NO”以避免将 LDAP 信息高速缓存在数据库、节点和 DCS 目录中。

## 搜索 LDAP 服务器

DB2 数据库系统搜索当前 LDAP 服务器，但是在具有多个 LDAP 服务器的环境中，可以定义搜索范围。

例如，如果在当前 LDAP 服务器中未找到信息，那么可以指定自动搜索其他所有 LDAP 服务器；也可以将搜索范围限制为仅搜索当前 LDAP 服务器或者仅搜索本地 DB2 数据库目录。

当设置搜索范围时，就会设置整个企业的缺省搜索范围。搜索范围由 DB2 数据库概要文件注册表变量 DB2LDAP_SEARCH_SCOPE 控制。要设置搜索范围值，在 db2set 命令中使用 -gl 选项，该选项表示“LDAP 中的全局”:

```
db2set -gl db2ldap_search_scope=<value>
```

可能的值包括: “local”、“domain”或“global”。如果未设置它，那么缺省值为“domain”，它将搜索范围限制为在当前 LDAP 服务器上的目录中搜索。

例如，在创建新的数据库之后可能需要将搜索范围初始设置为“global”。这就允许将任何 DB2 客户机配置为使用 LDAP 来搜索所有 LDAP 服务器以查找数据库。在每台客户机首次连接之后，一旦在每台机器上记录了该条目，如果启用了高速缓存，那么搜索范围可更改为“local”。一旦更改为“local”，每台客户机将不扫描任何 LDAP 服务器。

**注**: DB2 数据库概要文件注册表变量 DB2LDAP_KEEP_CONNECTION 和 DB2LDAP_SEARCH_SCOPE 是唯一可以在 LDAP 中全局级设置的注册表变量。

# 第 2 部分 数据库

© Copyright IBM Corp. 1993, 2010

# 第 6 章 数据库

DB2 数据库是*关系数据库*。*数据库*将所有数据存储在彼此相关的表中。在这些表之间建立关系，以便可以共享数据并使重复项最少。

*关系数据库*是被视为一组表并按照关系数据模型操作的数据库。它包含一组用来存储、管理和访问数据的对象。这种对象示例包括表、视图、索引、函数、触发器和程序包。对象可以由系统（系统定义的对象）或用户（用户定义的对象）定义。

*分布式关系数据库*包含一组表和其他对象，它们分布在不同但内部相连的计算机系统中。每个计算机系统都有一个关系数据库管理器，用于管理其环境中的表。数据库管理器相互间的通信和合作方式允许给定数据库管理器在另一个计算机系统上执行 SQL 语句。

*分区关系数据库*是在多个数据库分区中管理其数据的关系数据库。这种将数据分布在多个数据库分区中的方式对大多数 SQL 语句的用户来说是透明的。但是，某些数据定义语言（DDL）语句会考虑数据库分区信息，例如，CREATE DATABASE PARTITION GROUP。DDL 是用来描述数据库中的数据关系的 SQL 语句子集。

*联合数据库*是其数据存储在多个数据源（例如，不同的关系数据库）中的关系数据库。这些数据看起来就像都位于单个大型数据库中一样，并且可以通过传统 SQL 查询来访问。对数据所作的更改可以显式定向至适当的数据源。

## 设计数据库

设计数据库时，您其实是在对实际业务系统进行建模，该系统包含一组实体及其特征或*属性*，以及这些实体之间的规则或关系。

第一步是描述要表示的系统。例如，如果要为出版系统创建数据库，那么该系统应包含几种类型的实体，如书籍、作者、编辑和出版者。对于其中每个实体，您需要记录一些信息或属性:

- *书籍:* 标题、ISBN、出版日期、出版地、出版者....
- *作者:* 姓名、地址、电话号码和传真号码、电子邮件地址....
- *编辑:* 姓名、地址、电话号码和传真号码、电子邮件地址....
- *出版者:* 姓名、地址、电话号码和传真号码、电子邮件地址....

数据库不仅需要表示这些类型的实体及其属性，还需要一种使这些实体相互关联的方法。例如，需要表示书籍与作者之间的关系、书籍/作者与编辑之间的关系以及书籍/作者与出版者之间的关系。

数据库中的实体之间存在三种类型的关系:

**一对一关系**

在这种类型的关系中，实体的每个实例仅与另一个实体的一个实例相关。当前，在上面描述的场景中不存在一对一关系。

© Copyright IBM Corp. 1993, 2010

**一对多关系**

在这种类型的关系中，实体的每个实例与另一个实体的一个或多个实例相关。例如，一个作者可能写了多本书籍，但某些书籍只有一个作者。这是关系数据库中建模的最常见的关系类型。

**多对多关系**

在这种类型的关系中，给定实体的许多实例与另一个实体的一个或多个实例相关。例如，合著者可以写许多本书籍。

因为数据库由表组成，所以需要构造一组能最好地保存这些数据的表，并且表中的每个单元格保存单个视图。有许多种可能的方法来执行此任务。作为数据库设计者，您的工作就是提出一组可能最好的表。

例如，可以创建包含许多行和列的单个表来保存所有信息。但是，使用此方法时，某些信息将会重复。其次，数据输入和数据维护将耗费大量时间，并且容易出错。与此单个表设计相比，*关系数据库*允许您使用多个简单的表，从而减少冗余并避免一个不容易管理的大型表所产生的困难。在关系数据库中，表应该包含关于单种类型的实体的信息。

另外，由于多个用户访问和更改数据，所以必须维护关系数据库数据完整性。每当共享数据时，就需要确保数据库表中的值准确。

您可以:

- 使用隔离级别来确定访问数据时如何锁定数据或者将该数据与其他进程隔离开。
- 通过定义约束来强制实施业务规则，从而保护数据和定义数据之间的关系。
- 创建触发器以执行复杂的跨表数据验证。
- 实现恢复策略来保护数据，以便可以将该数据复原到一致的状态。

数据库设计任务远比此处说明的要复杂得多，它需要考虑许多因素，比如，空间需求、键、索引、约束、安全性和权限等。您可以在 DB2 信息中心以及关于此主题的许多零售 DB2 书籍中找到一些这方面的信息。

## 数据库目录和文件

当创建一个数据库时，关于该数据库的信息（包括缺省信息）会存储在目录层次结构中。

此分层目录结构的创建位置取决于您在 CREATE DATABASE 命令中提供的信息。如果在创建数据库时未指定目录路径或驱动器的位置，那么将使用缺省位置。建议您明确指出希望在何处创建数据库。

在 CREATE DATABASE 命令中指定为数据库路径的目录中，将创建一个使用实例名的子目录。这个子目录确保在同一目录下的不同实例中创建的数据库不会使用相同的路径。在实例名子目录下面，将创建一个名为 NODE0000 的子目录。这个子目录可以区分逻辑分区数据库环境中的数据库分区。在节点名目录下面，将创建一个名为 SQL00001 的子目录。此子目录的名称使用了数据库标记并表示正在创建的数据库。SQL00001 包含与创建的第一个数据库以及随后创建的具有更高编号（SQL00002 等）的数据库相关联的对象。这些子目录可以区分在 CREATE DATABASE 命令中指定的目录下的实例中创建的数据库。

显示的目录结构为如下所示: *your_database_path*/*your_instance*/NODE0000/SQL00001/。

数据库目录中包含下列作为 CREATE DATABASE 命令的一部分进行创建的文件。

- SQLBP.1 和 SQLBP.2 文件中都包含缓冲池信息。这两个文件互为副本以实现备份。
- SQLSPCS.1 和 SQLSPCS.2 文件中都包含表空间信息。这两个文件互为副本以实现备份。
- SQLSGF.1 和 SQLSGF.2 文件中都包含与数据库的自动存储器相关的存储路径信息。这两个文件互为副本以实现备份。
- SQLDBCONF 文件中包含数据库配置信息。切勿编辑此文件。

  **注**: SQLDBCON 文件用于先前发行版，它包含可以在 SQLDBCONF 被破坏时使用的类似信息。
  要更改配置参数，请使用 UPDATE DATABASE CONFIGURATION 和 RESET DATABASE CONFIGURATION 命令。
- DB2RHIST.ASC 历史记录文件及其备份 DB2RHIST.BAK 中包含关于备份、复原、表装入、表重组、表空间改变和其他数据库更改的历史记录信息。

  DB2TSCHG.HIS 文件中包含日志文件级别的表空间更改的历史记录。对于每个日志文件，DB2TSCHG.HIS 中包含有助于确定日志文件影响哪些表空间的信息。表空间恢复使用此文件中的信息来确定在进行表空间恢复期间要处理哪些日志文件。可以在文本编辑器中检查这两个历史记录文件中的内容。
- 日志控制文件 SQLOGCTL.LFH.1 及其镜像副本 SQLOGCTL.LFH.2 和 SQLOGMIR.LFH 中包含有关活动日志的信息。

  恢复处理过程使用这些文件中的信息来确定要在日志中后退多远来开始恢复。SQLOGDIR 子目录中包含实际的日志文件。

  **注**: 您应确保不要将日志子目录映射到用于存储数据的磁盘。这样，在磁盘发生问题时，只会影响到数据或日志，而不会同时影响这两者。由于日志文件与数据库容器不会争用同一磁盘磁头的移动，因此这可提供很多性能方面的好处。要更改日志子目录的位置，请更改 **newlogpath** 数据库配置参数。
- SQLINSLK 文件用于确保一个数据库只能由数据库管理器的一个实例使用。

在创建数据库的同时，还创建了详细死锁事件监视器。详细死锁事件监视器文件存储在目录节点的数据库目录中。当事件监视器达到它要输出的最大文件数时，它将取消激活，并且将把一条消息写入通知日志中。这样可防止事件监视器消耗过多的磁盘空间。除去不再需要的输出文件将允许在下一次数据库激活时再次激活事件监视器。

## 非自动存储器数据库中的 SMS 数据库目录的其他信息

在非自动存储器数据库中，SQLT* 子目录包含运作数据库所需的缺省“系统管理的空间”（SMS）表空间。将创建三个缺省表空间:

- SQLT0000.0 子目录中包含带有系统目录表的目录表空间。
- SQLT0001.0 子目录中包含缺省临时表空间。
- SQLT0002.0 子目录中包含缺省用户数据表空间。

每个子目录或容器中都会创建一个名为 SQLTAG.NAM 的文件。这个文件可以标记正在使用中的子目录，因此在以后创建其他表空间时，不会尝试使用这些子目录。

此外，名为 SQL*.DAT 的文件中还存储有关子目录或容器包含的每个表的信息。星号（*）将被唯一的一组数字取代，用来识别每个表。对于每个 SQL*.DAT 文件，可能有一个或多个下列文件，这取决于表类型、表的重组状态或者表是否存在索引、LOB 或 LONG 字段:

- SQL*.BKM（如果它是一个 MDC 表，那么它包含块分配信息）
- SQL*.LF（包含 LONG VARCHAR 或 LONG VARGRAPHIC 数据）
- SQL*.LB（包含 BLOB、CLOB 或 DBCLOB 数据）
- SQL*.XDA（包含 XML 数据）
- SQL*.LBA（包含有关 SQL*.LB 文件的分配和可用空间信息）
- SQL*.INX（包含索引表数据）
- SQL*.IN1（包含索引表数据）
- SQL*.DTR（包含用于重组 SQL*.DAT 文件的临时数据）
- SQL*.LFR（包含用于重组 SQL*.LF 文件的临时数据）
- SQL*.RLB（包含用于重组 SQL*.LB 文件的临时数据）
- SQL*.RBA（包含用于重组 SQL*.LBA 文件的临时数据）

**注:** 继续支持 LONG VARCHAR 和 LONG VARGRAPHIC 数据类型，但不推荐使用这两种类型，将来的发行版中可能会将其除去。

## 数据库配置文件

为每个数据库创建*数据库配置文件*。在版本 8.2 之前，此文件称为 SQLDBCON，而在版本 8.2 和更高版本中则称为 SQLDBCONF。系统已为您创建此文件。

此文件包含影响数据库使用的各种*配置参数*的值，如:

- 在创建数据库时指定或使用的参数（例如，数据库代码页、整理顺序和 DB2 数据库发行版级别）
- 指示数据库当前状态的参数（例如，备份暂挂标志、数据库一致性标志和前滚暂挂标志）
- 定义数据库操作可使用的系统资源数量的参数（例如，缓冲池大小、数据库日志记录和排序内存大小）。

**注:** 如果您使用非 DB2 数据库管理器提供的方法来编辑 db2system、SQLDBCON（在版本 8.2 之前）或 SQLDBCONF（在版本 8.2 和更高版本中）文件，那么可能会使数据库不可用。因此，不要使用非数据库管理器记载和支持的方法来更改这些文件。

**性能提示:** 许多配置参数都有缺省值，但可能需要更新以获取数据库最佳性能。缺省情况下，在执行 CREATE DATABASE 命令的过程中会调用“配置顾问程序”，以便已为您的环境配置某些参数的初始值。

**对于多分区数据库:** 当数据库分布在多个数据库分区时，配置文件在所有数据库分区上都应该相同。一致性是必需的，因为查询编译器根据本地节点配置文件中的信息来编译分布的 SQL 语句，并创建一个访问方案以满足 SQL 语句的需要。维护数据库分区上的不同配置文件可能产生不同的访问方案，这取决于预编译该语句所在的数据库分区。

## 节点目录

数据库管理器在编目第一个数据库分区时会创建*节点目录*。

要编目数据库分区，可使用 CATALOG NODE 命令。要列示本地节点目录的内容，可使用 LIST NODE DIRECTORY 命令。

在每个数据库客户机上都创建并维护节点目录。对于具有客户机可以访问的一个或多个数据库的每个远程工作站，该目录都包含一个条目。无论何时请求数据库连接或实例连接，DB2 客户机都会使用该节点目录中的通信端点信息。

该目录中的条目还包含客户机与远程数据库分区通信要使用的通信协议的类型信息。编目本地数据库分区会为位于同一台计算机上的实例创建一个别名。

## 本地数据库目录

*本地数据库目录*文件存在于定义了数据库的每条路径（或 Windows 操作系统的“驱动器”）中。对于可从该位置访问的每个数据库，此目录中都包含一个相应的条目。

每一个条目包含:

- 随 CREATE DATABASE 命令提供的数据库名称
- 数据库别名（如果未指定别名，它与数据库名称相同）
- 随 CREATE DATABASE 命令提供的描述该数据库的注释
- 该数据库的根目录的名称
- 其他系统信息

## 系统数据库目录

对于数据库管理器的每个实例，都存在一个*系统数据库目录*文件，该文件对于针对此实例编目的每个数据库都包含一个条目。

当发出 CREATE DATABASE 命令时，将隐式地对数据库进行编目，也可以使用 CATALOG DATABASE 命令显式地编目该数据库。

对于创建的每个数据库，都要将包含下列信息的一个条目添加至该目录:

- 随 CREATE DATABASE 命令提供的数据库名称
- 数据库别名（如果未指定别名，它与数据库名称相同）
- 随 CREATE DATABASE 命令提供的数据库注释
- 本地数据库目录的位置
- 指示该数据库是*间接*数据库的指示符，表示数据库位于当前数据库管理器实例上
- 其他系统信息

在 UNIX 平台和分区数据库环境中，必须确保所有数据库分区始终访问该实例主目录的 `sqldbdir` 子目录中的同一系统数据库目录文件 `sqldbdir`。如果系统数据库目录或同一 `sqldbdir` 子目录中的系统意向文件 `sqldbins` 是指向共享文件系统中的另一个文件的符号链接，可能会发生不可预测的错误。

## 创建节点配置文件

如果数据库要在分区数据库环境中运行，那么必须创建一个名为 `db2nodes.cfg` 的节点配置文件。

要启用数据库分区，db2nodes.cfg 文件必须位于实例主目录的 sqllib 子目录中，这样才能启动数据库管理器。此文件包含一个实例中所有数据库分区的配置信息，并且它由该实例的所有数据库分区共享。

### Windows 注意事项

如果正在 Windows 上使用 DB2 Enterprise Server Edition，那么将在创建实例时创建节点配置文件。您不应尝试手动创建或修改节点配置文件。可使用 db2ncrt 命令来将数据库分区服务器添加至实例。可使用 db2ndrop 命令从实例中删除数据库分区服务器。可使用 db2nchg 命令来修改数据库分区服务器配置，包括将数据库分区服务器从一台计算机移至另一台计算机；更改 TCP/IP 主机名；或选择另一逻辑端口或网络名。

**注:** 不应该在不是数据库管理器创建的 sqllib 子目录下创建文件或目录，以防止删除实例时丢失数据。但有两个例外情况。如果系统支持存储过程，那么将该存储过程应用程序放入 sqllib 子目录下的 function 子目录中。另一个例外是在已创建用户定义的函数（UDF）的情况下。允许 UDF 可执行程序位于同一个目录中。

对于属于一个实例的每个数据库分区该文件都包含一行。每行的格式如下:

*dbpartitionnum hostname* [*logical-port* [*netname*]]

记号由空格定界。这些变量是:

*dbpartitionnum*

数据库分区号唯一地定义数据库分区，可在 0 到 999 之间。数据库分区号必须以升序顺序排序。该顺序中可以有间隔。

一旦指定了数据库分区号，就不能对其进行更改。否则，分布图（它指定数据分发方式）中的信息可能不正确。

如果删除一个数据库分区，那么它的数据库分区号可以再次用于添加的任何新数据库分区。

数据库分区号用于在数据库目录中生成数据库分区名。它的格式为:

```
NODE nnnn
```

*nnnn* 是数据库分区号，其左边以零填充。CREATE DATABASE 和 DROP DATABASE 命令也使用此数据库分区号。

*hostname*

用作分区间通信的 IP 地址的主机名。使用主机名的标准名称。/etc/hosts 文件也应该使用标准名称。如果未在 db2nodes.cfg 文件和 /etc/hosts 文件中使用标准名称，那么可能接收到错误消息“SQL30082N RC=3”。

（指定 netname 时例外。在此情况下，netname 用于大多数通信，而 hostname 仅用于 db2start、db2stop 和 db2_all。）

*logical-port*

此参数是可选的，它指定该数据库分区的逻辑端口号。此号码与数据库管理器实例名一起用来标识 etc/services 文件中的 TCP/IP 服务名称条目。

IP 地址和逻辑端口的组合被用作熟知地址，且在所有支持数据库分区间通信连接的应用程序中必须是唯一的。

对于每个主机名，一个*逻辑端口*必须为 0（零）或空白（缺省为 0）。与此*逻辑端口*相关联的数据库分区是与客户机连接的主机上的缺省节点。可以使用 db2profile 脚本中的 **DB2NODE** 环境变量或 sqlesetc() API 来覆盖它。

*netname*

此参数是可选的，并且用于支持有多个活动 TCP/IP 接口的主机，每个接口有其自己的主机名。

以下示例显示了一个 RS/6000® SP 系统的可能节点配置文件，在该系统上，SP2EN1 有多个 TCP/IP 接口和两个逻辑分区，并且使用 SP2SW1 作为 DB2 数据库接口。此示例还显示了从 1 开始（而不是从 0 开始）的数据库分区号以及 *dbpartitionnum* 序列中的间隙:

*表 39. 数据库分区号示例表。*

| *dbpartitionnum* | *hostname* | *logical-port* | *netname* |
|---|---|---|---|
| 1 | SP2EN1.mach1.xxx.com | 0 | SP2SW1 |
| 2 | SP2EN1.mach1.xxx.com | 1 | SP2SW1 |
| 4 | SP2EN2.mach1.xxx.com | 0 | |
| 5 | SP2EN3.mach1.xxx.com | | |

可以使用选择的编辑器更新 `db2nodes.cfg` 文件。（例外情况: 不应在 Windows 上使用编辑器）。但是，必须小心保护文件中信息的完整性，这是因为数据库分区要求当您发出 db2start 时使节点配置文件处于锁定状态，而在发出 db2stop 来结束数据库管理器之后又将它解锁。文件被锁定时，db2start 命令可在必要时更新该文件。例如，可发出 db2start 并指定 **RESTART** 选项或 **ADD DBPARTITIONNUM** 选项。

**注**: 如果 db2stop 命令不成功而又未解锁该节点配置文件，那么发出 db2stop **FORCE** 来将其解锁。

## 更改节点和数据库配置文件

要更新数据库配置文件，请运行带有适当选项的 AUTOCONFIGURE 命令。

“配置顾问程序”通过建议修改某些配置参数并为它们提供建议值来帮助调整性能和平衡每个实例中单个数据库的内存需求。

如果打算更改任何数据库分区组（添加或删除数据库分区或者除去现有数据库分区），那么必须更新节点配置文件。如果打算更改数据库，那么应查看配置参数的值。在根据数据库的使用方式更改数据库的过程中，可以定期调整某些值。

**注**: 如果修改任何参数，那么在发生下列各种情况之前，不更新值:

- 对于数据库参数，在与所有应用程序断开连接之后，与数据库建立第一个新连接时
- 对于数据库管理器参数，下一次停止和启动该实例时

在大多数情况下，“配置顾问程序”所建议的值将比缺省值提供更好的性能，因为它们是根据有关工作负载和您自己的特定服务器的信息确定的。但是，这些值是为提高数据库系统的性能而设计的，并不一定能优化该系统。应该将这些值当作一个起始点，然后进一步调整以获得优化的性能。

在版本 9.1 中，创建数据库时会自动调用“配置顾问程序”。要禁用此功能或显式启用它，可在创建数据库之前使用 db2set 命令来实现。示例:

```
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=YES
```

有关缺省情况下启用的其他功能，请参阅第 17 页的『自动功能』。

要从命令行使用“配置顾问程序”，可使用 AUTOCONFIGURE 命令。

要使用命令行来更新数据库管理器配置中的个别参数，请输入:

```
UPDATE DBM CFG USING <config_keyword>=<value>
```

要使用命令行来更新数据库配置中的个别参数，请输入:

```
UPDATE DB CFG FOR <database_alias>
   USING <config_keyword>=<value>
```

可以在单个命令中更新一个或多个 <config_keyword>=<value> 组合。对数据库管理器配置文件的大多数更改只有在将它们装入内存之后才会生效。对于服务器配置参数，这在运行 START DATABASE MANAGER 命令期间发生。对于客户机配置参数，这在重新启动应用程序时发生。

要查看或打印当前数据库管理器配置参数，可使用 GET DATABASE MANAGER CONFIGURATION 命令。

要从客户机应用程序中访问配置顾问程序，请调用 db2AutoConfig API。要从客户机应用程序中更新数据库管理器配置或数据库配置文件中的各个参数，请调用 db2CfgSet API。

### 数据库恢复日志

*数据库恢复日志*保存对一个数据库所做的所有更改（包括新表的添加或对现有表的更新）的记录。

此日志由大量*日志扩展数据块*组成，每一个日志扩展数据块包含在一个称为*日志文件*的单独文件中。

数据库恢复日志可以用于确保故障（例如，系统断电或应用程序出错）不会使数据库处于不一致的状态。如果发生故障，那么回滚已进行但未落实的更改，并重新执行所有可能未实际写入磁盘的已落实事务。这些操作确保了数据库的完整性。

## 数据库对象的空间需求

数据库对象的大小估计不可能做到很精确。磁盘碎片、可用空间以及使用变长列所造成的开销都使大小估计变得十分困难，因为可能的列类型和行长度的范围实在是太广了。

在最初估计数据库大小后，将创建一个测试数据库，并用有代表性的数据对其进行填充。然后，使用 db2look 实用程序来生成数据库的数据定义语句。

估计数据库的大小时，必须考虑下列各项的影响:

- 系统目录表
- 用户表数据
- 长字段（LF）数据
- 大对象（LOB）数据

- 索引空间
- 日志文件空间
- 临时工作空间

还应考虑下列各项的开销和空间需求:

- 本地数据库目录文件
- 系统数据库目录文件
- 操作系统所需的文件管理开销，包括:
  - 文件块大小
  - 目录控制空间

## 日志文件的空间需求

日志控制文件需要 56 KB 的空间。

至少还需要足够的空间以进行活动日志配置，可进行如下计算

```
(logprimary + logsecond) * (logfilsiz + 2 ) * 4096
```

其中:

- *logprimary* 是在数据库配置文件中定义的主日志文件数
- *logsecond* 是在数据库配置文件中定义的辅助日志文件的数目；在此计算中，不能将 *logsecond* 设置为 *-1*。（当 *logsecond* 设置为 -1 时，您是在请求一个无限活动日志空间。）
- *logfilsiz* 是在数据库配置文件中定义的每个日志文件中的页数
- 2 是每个日志文件所需的标题页的数目
- 4096 是一页中的字节数

如果已对数据库启用了循环日志记录，那么此公式的结果是将为日志记录分配的所有空间；也就是说，不会分配更多空间，并且对于任何日志文件，您不会接收到“磁盘空间不足”错误。

若允许对该数据库执行前滚恢复，应该考虑特殊的日志空间需求:

- 当 *logarchmeth1* 配置参数设置为 logretain 时，日志文件将被归档在日志路径目录中。除非将日志文件移至另一个位置，否则，联机磁盘空间最终将会填满。
- 当 *logarchmeth1* 配置参数设置为 userexit、DISK 或 VENDOR 时，用户出口程序会将已归档的日志文件移至另一个位置。要允许下列情况，附加的日志空间仍是必需的:
  - 等待用户出口程序移动的联机归档日志
  - 格式化新的日志文件，以供将来使用

如果已对数据库启用无限日志记录（即，将 *logsecond* 设置为 *-1*），那么必须将 *logarchmeth1* 配置参数设置为除 OFF 或 LOGRETAIN 之外的值，以便启用归档日志记录。数据库管理器将在日志路径中至少保留 *logprimary* 所指定的数目的活动日志文件，所以上面公式中 *logsecond* 的值不应使用 -1。确保提供额外磁盘空间以允许归档日志文件导致的延迟。

如果正在镜像日志路径，那么需要将估计的日志文件空间需求增大一倍。

## 轻量级目录访问协议（LDAP）目录服务

目录服务是一个关于分布式环境中的多个系统和服务的资源信息的存储库；它提供对这些资源的客户机和服务器访问。

客户机和服务器将使用目录服务来找出如何访问其他资源。在分布式环境中，必须将有关这些其他资源的信息输入到目录服务存储库中。

*轻量级目录访问协议（LDAP）*是业界标准的目录服务访问方法。每个数据库服务器实例都会将它的存在情况发布给 LDAP 服务器，并在创建数据库时向 LDAP 目录提供数据库信息。客户机与数据库连接后，就可以从 LDAP 目录检索服务器的目录信息。不再要求每个客户机将目录信息以本地方式存储在每台计算机上。客户机应用程序搜索 LDAP 目录以找出连接数据库所需的信息。

**注:** 将数据库服务器实例发布至 LDAP 服务器不是一个自动过程，而是必须由管理员手动完成。

作为 DB2 系统的管理员，您可以建立并维护目录服务。“配置助手”可帮助维护此目录服务。可通过“轻量级目录访问协议”（LDAP）目录服务来使目录服务对 DB2 数据库管理器可用。要使用 LDAP 目录服务，首先必须有一个 DB2 数据库管理器支持的 LDAP 服务器，以便存储目录信息。

**注:** 在 Windows 域环境中运行时，LDAP 服务器已经可用，因为它与 Windows Active Directory 集成在一起。因此，每台运行 Windows 的计算机都可使用 LDAP。

如果企业环境中有大量客户机，并且在每台客户机上更新本地目录非常困难，那么 LDAP 目录在这种环境中非常有用。在这种情况下，应考虑将目录条目存储在 LDAP 服务器中，以便在一个位置（即 LDAP 服务器中）完成目录条目的维护。

## 创建数据库

可以使用 CREATE DATABASE 命令来创建数据库。要从客户机应用程序中创建数据库，请调用 sqlecrea API。

### 开始之前

创建数据库之前，应花足够多的时间来设计数据库的内容、布局、潜在增长和用途。

下列数据库特权被自动授予 PUBLIC：对系统目录视图的 CREATETAB、BINDADD、CONNECT、IMPLICIT_SCHEMA 和 SELECT。但是，如果存在 **RESTRICTIVE** 选项，那么不会自动对 PUBLIC 授予任何特权。有关 **RESTRICTIVE** 选项的更多信息，请参阅 CREATE DATABASE 命令。

### 关于本任务

创建数据库时，为您完成了下列所有任务:

- 设置数据库所需的所有系统目录表
- 分配数据库恢复日志
- 创建数据库配置文件并设置缺省值
- 将数据库实用程序与数据库绑定

### 过程

- 要从客户机应用程序中创建数据库，请调用 sqlecrea API。
- 要使用命令行处理器来创建数据库，请发出 CREATE DATABASE 命令。

  例如，以下命令在缺省位置创建一个名为 PERSON1 的数据库，并带有相关注释 ″Personnel DB for BSchiefer Co″。

```
CREATE DATABASE personl
        WITH "Personnel DB for BSchiefer Co"
```

### 下一步操作

#### 配置顾问程序

“配置顾问程序”通过建议修改某些配置参数并为它们提供建议值来帮助调整性能和平衡每个实例中单个数据库的内存需求。创建数据库时，缺省情况下会自动调用“配置顾问程序”。

可以覆盖此缺省值，以便不会使用下列其中一种方法来自动调用配置顾问程序:

- 发出 CREATE DATABASE 命令并指定 **AUTOCONFIGURE APPLY NONE** 参数。
- 将 **DB2_ENABLE_AUTOCONFIG_DEFAULT** 注册表变量设置为 NO:

  ```
  db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
  ```

  但是，如果您指定 **AUTOCONFIGURE** 参数，那么将忽略此注册表变量的设置。

请参阅第 17 页的『自动功能』，以了解当您创建数据库时缺省情况下启用的其他功能。

#### 事件监视器

在创建数据库的同时，还创建了详细死锁事件监视器。同其他监视器一样，这个事件监视器将造成一些开销。如不需要详细死锁事件监视器，可运行以下命令来删除:

```
DROP EVENT MONITOR db2detaildeadlock
```

要限制此事件监视器消耗的磁盘空间，那么在输出文件达到最大数时取消激活事件监视器，并将此消息写入到管理通知日志。将不再需要的输出文件除去即可在下次激活数据库时再次激活事件监视器。

#### 远程数据库

您可以在另一个可能是远程的实例中创建数据库。要在另一个（远程）数据库分区服务器中创建数据库，首先必须连接至该服务器。在处理期间，以下命令临时建立了数据库连接:

```
CREATE DATABASE database_name AT DBPARTITIONNUM options
```

在这种类型的环境中，还可以对不同于缺省实例的实例（包括远程实例）执行实例级管理。有关如何执行此操作的指示信息，请参阅 db2iupdt（更新实例）命令。

#### 数据库代码页

缺省情况下，使用 UTF-8（Unicode）代码集创建数据库。

要覆盖数据库的缺省代码页，需要在创建数据库时指定想要的代码集和地域。有关设置代码集和地域的信息，请参阅 CREATE DATABASE 命令或 sqlecrea API。

## 自动存储器数据库

缺省情况下，数据库管理器会将所有数据库都作为“自动存储器”数据库来创建。要创建不是“自动存储器”数据库的数据库，请在发出 CREATE DATABASE 命令时指定 AUTOMATIC STORAGE NO。

启用自动存储器数据库有一套一个或多个存储路径与之关联。可以将表空间定义为*由自动存储器管理*，而它的容器由数据库管理器根据那些存储路径来指定和分配。

仅当创建数据库时才能对该数据库启用自动存储器。同样，不能对最初设计为使用自动存储器的数据库禁用自动存储器。

缺省情况下，将所有数据库作为自动存储器数据库来创建。要创建不是自动存储器数据库的数据库，请在发出 CREATE DATABASE 命令时指定 **AUTOMATIC STORAGE NO**。

以下是禁用自动存储器的一些示例:

```
CREATE DATABASE ASNODB1 AUTOMATIC STORAGE NO
CREATE DATABASE ASNODB2 AUTOMATIC STORAGE NO ON X:
```

显式或隐式启用自动存储器的一些示例:

```
CREATE DATABASE DB1
CREATE DATABASE DB2 AUTOMATIC STORAGE YES ON X:
CREATE DATABASE DB3 ON /data/path1, /data/path2
CREATE DATABASE DB4 ON D:\StoragePath DBPATH ON C:
```

根据所用的语法，数据库管理器抽取下列有关存储位置的两部分信息:

- 数据库路径（数据库管理器将数据库的各种控制文件存储在其中）:
  - 如果您指定 **DBPATH ON**，那么这指示数据库路径。
  - 如果不指定 **DBPATH ON**，那么 **ON** 中列示的第一个路径指示数据库路径（和存储路径）。
  - 如果既未指定 **DBPATH ON** 又未指定 **ON**，那么使用 **dftdbpath** 数据库管理器配置参数来确定数据库路径。
- 存储路径（数据库管理器在其中创建自动存储器表空间容器）:
  - 如果指定了 **ON**，那么列示的所有路径都是存储路径。
  - 如果未指定 **ON**，那么将存在单个存储路径，它被设置为 **dftdbpath** 数据库管理器配置参数的值。

对于前面显示的示例，下表概述了所使用的数据库路径和存储路径:

*表 40. 自动存储器数据库路径和存储路径*

| **CREATE DATABASE 命令** | **数据库路径** | **存储路径** |
|---|---|---|
| CREATE DATABASE DB1 AUTOMATIC STORAGE YES | **dftdbpath** 配置参数的值 | **dftdbpath** 配置参数的值 |
| CREATE DATABASE DB2 AUTOMATIC STORAGE YES ON X: | X: | X: |
| CREATE DATABASE DB3 ON /data/path1, /data/path2 | /data/path1 | /data/path1, /data/path2 |

*表 40. 自动存储器数据库路径和存储路径 (续)*

| CREATE DATABASE 命令 | 数据库路径 | 存储路径 |
|---|---|---|
| CREATE DATABASE DB4 ON D:\StoragePath DBPATH ON C: | C: | D:\StoragePath |

提供的存储路径必须存在并且可以访问。在分区数据库环境中，将在每个数据库分区上使用相同的存储路径。除非将数据库分区表达式用作存储路径名的一部分，否则不能为特定数据库分区指定一组唯一的存储路径。将数据库分区表达式用作存储路径名的一部分允许在存储路径中反映数据库分区号，这样生成的路径名在每个数据库分区上都不相同。

使用自变量 $N（即，在 $N 前面添加一个空格）来指示数据库分区表达式。可以在存储路径中的任何位置使用数据库分区表达式，并且可以指定多个数据库分区表达式。使用空格字符来表示数据库分区表达式的结束区表达式以空格字符结束；在对数据库分区表达式求值以后，该空格后跟的所有内容都将追加至存储路径。如果存储路径中的数据库分区表达式后面没有空格字符，那么假定其余字符串是该表达式的一部分。下表列示了 $N 自变量唯一有效的格式。从左向右按运算符求值，% 表示模数运算符。示例中的数据库分区号为 10。

*表 41. 数据库分区表达式*

| 语法 | 示例 | 值 |
|---|---|---|
| [blank]$N | " $N" | 10 |
| [blank]$N+[number] | " $N+100" | 110 |
| [blank]$N%[number] | " $N%5" | 0 |
| [blank]$N+[number]%[number] | " $N+1%5" | 1 |
| [blank]$N%[number]+[number] | " $N%4+2" | 4 |

以下是使用数据库分区表达式的示例:

```
CREATE DATABASE TESTDB ON "/path1ForNode $N",
  "/path2ForNode $N" DBPATH ON "/dbpathForNodes"
```

以下是嵌入在路径中的数据库分区表达式的示例:

```
CREATE DATABASE TESTDB ON "/path1ForNode $N",
  "/path2ForNode $N suffix" DBPATH ON "/dbpathForNodes"
```

**注**: 无论数据库分区表达式是在 **DBPATH ON** 中显式指定的，还是通过在第一个存储路径中使用数据库分区表达式隐式指定的，数据库分区表达式在数据库路径中都无效。

计算出给定数据库分区的存储路径的可用空间之后，数据库管理器将检查该存储路径中是否存在下列目录或安装点，并使用找到的第一个目录或安装点:

```
storage path/instance name/NODE####/database name
storage path/instance name/NODE####
   storage path/instance name
storage path
```

其中:

*storage path*
是与数据库相关联的存储路径

*instance name*
是数据库所在的实例

**NODE####**
是数据库分区号（例如，NODE0000 或 NODE0001）

*database name*
是数据库的名称

可以将文件系统安装在存储路径中的某个位置，并且数据库管理器将意识到可用于表空间容器的实际可用空间大小可能和与存储路径目录本身相关联的那个大小不同。

请考虑以下示例：在一台物理计算机上存在两个逻辑数据库分区，并且有一个存储路径：`/db2data`。每个数据库分区都可以使用此存储路径，但是您可能想通过为每个数据库分区创建单独的文件系统来将每个数据库分区中的数据分隔开。文件系统安装在 `/db2data/instance/NODE####` 中。在存储路径中创建容器并确定可用空间时，数据库管理器不会检索 `/db2data` 的可用空间信息，但是会检索相应 `/db2data/instance/NODE####` 目录的可用空间信息。

每当创建数据库时都会创建三个缺省表空间。如果您不提供显式表空间定义作为 CREATE DATABASE 命令的一部分，那么会将表空间创建为自动存储器表空间。

创建数据库之后，可以使用 ALTER DATABASE 语句的 ADD STORAGE 子句来对该数据库添加新的存储路径，如以下示例中所示：

```
ALTER DATABASE ADD STORAGE ON '/data/path3', '/data/path4'
```

## 自动存储器限制

在决定是否创建使用自动存储器的数据库时，应考虑一些限制。

- 创建数据库之后，不能对它禁用或启用自动存储器。
- 存储路径必须是绝对路径名。它们可以是 Windows 操作系统上的路径或盘符。数据库路径必须是盘符。 最大路径长度是 175 个字符。
- 对于分区数据库，必须在每个数据库分区上使用同一组存储路径（除非您使用数据库分区表达式）。
- 无论数据库分区表达式是使用 CREATE DATABASE 命令的 **DBPATH ON** 选项显式指定的，还是通过在第一个存储路径中使用数据库分区表达式隐式指定的，数据库分区表达式在数据库路径中都无效。

## 将自动存储路径添加至启用了自动存储器的数据库

通过使用 ALTER DATABASE 语句，可以将自动存储路径添加至启用了自动存储器的数据库。仅当创建数据库时才能对该数据库启用自动存储器。

对多分区数据库环境添加存储路径时，每个数据库分区上都必须存在存储路径。如果指定的路径在每个数据库分区上都不存在，那么将回滚该语句。

要将存储路径添加至现有数据库，请发出以下 ALTER DATABASE 语句：

```
ALTER DATABASE PATH pathname
```

## 监视存储器路径

数据库快照包括与数据库相关联的存储器路径列表。

如果自动存储器路径数为 0，那么不会对数据库启用自动存储器:

```
自动存储器路径数                    = ##
   自动存储器路径                     = <1st path>
   自动存储器路径                     = <2nd path>
...
```

如果缓冲池监视开关打开，那么还设置下列元素:

```
文件系统标识                              = 12345
文件系统可用空间（以字节计）              = 20000000000
文件系统已使用空间（以字节计）            = 40000000000000
文件系统总空间（以字节计）                = 40020000000000
```

将针对每个路径设置此数据：在单个数据库分区系统上，针对每个路径设置；在多分区数据库环境中，针对每个数据库分区设置。

另外，会在表空间快照中设置以下信息。此信息指示是否将表空间创建为自动存储器表空间:

```
使用自动存储器                        = Yes 或 No
```

## 复原数据库的含义

使用 RESTORE DATABASE 命令来从备份映像复原数据库。

在复原操作期间，可以选择数据库路径的位置，也可以重新定义与数据库相关联的存储器路径。通过将 TO、ON 和 DBPATH ON 的组合与 RESTORE DATABASE 命令配合使用来设置数据库路径和存储路径。

例如，以下是一些对于启用了自动存储器的数据库有效的 RESTORE 命令:

```
RESTORE DATABASE TEST1
RESTORE DATABASE TEST2 TO X:
RESTORE DATABASE TEST3 DBPATH ON D:
RESTORE DATABASE TEST3 ON /path1, /path2, /path3
RESTORE DATABASE TEST4 ON E:\newpath1, F:\newpath2 DBPATH ON D:
```

与数据库管理器在 CREATE DATABASE 命令中执行的操作一样，数据库管理器抽取下列有关存储位置的两部分信息:

- **数据库路径**（这是数据库管理器存储数据库的各种控制文件的位置）
  - 如果指定了 TO 或 DBPATH ON，那么这指示数据库路径。
  - 如果使用了 ON 但未对它指定 DBPATH ON，那么将随 ON 列示的第一个路径用作数据库路径（而不仅仅是存储路径）。
  - 如果没有指定 TO、ON 或 DBPATH ON 中的任何一个，那么 *dftdbpath* 数据库管理器配置参数将确定数据库路径。

  **注**: 如果磁盘上存在同名的数据库，那么会忽略数据库路径，而会将数据库放置在现有数据库所在的位置。
- **存储路径**（这是数据库管理器创建自动存储器表空间容器的位置）
  - 如果指定了 ON，那么列示的所有路径都认为是存储路径，并且将使用这些路径而不是存储在备份映像里的路径。

– 如果未指定 ON，那么不会对存储路径作出更改（会保留存储在备份映像中的存储路径）。

为了使这个概念更加清楚，在下表中显示了前面提到的 5 个 RESTORE 命令示例及其相应的存储路径:

*表 42. 与数据库路径和存储路径有关的复原含义*

| RESTORE DATABASE 命令 | 磁盘上不存在同名的数据库 | | 磁盘上存在同名的数据库 | |
|---|---|---|---|---|
| | 数据库路径 | 存储路径 | 数据库路径 | 存储路径 |
| RESTORE DATABASE TEST1 | *<dftdbpath>* | 使用在备份映像中定义的存储路径 | 使用现有数据库的数据库路径 | 使用在备份映像中定义的存储路径 |
| RESTORE DATABASE TEST2 TO X: | X: | 使用在备份映像中定义的存储路径 | 使用现有数据库的数据库路径 | 使用在备份映像中定义的存储路径 |
| RESTORE DATABASE TEST3<br>DBPATH ON /db2/databases | /db2/databases | 使用在备份映像中定义的存储路径 | 使用现有数据库的数据库路径 | 使用在备份映像中定义的存储路径 |
| RESTORE DATABASE TEST4<br>ON /path1, /path2, /path3 | /path1 | /path1, /path2, /path3 | 使用现有数据库的数据库路径 | /path1, /path2, /path3 |
| RESTORE DATABASE TEST5<br>ON E:\newpath1, F:\newpath2<br>DBPATH ON D: | D: | E:\newpath1, F:\newpath2 | 使用现有数据库的数据库路径 | E:\newpath1, F:\newpath2 |

对于已将存储路径定义为复原操作的一部分的那些情况，定义为使用自动存储器的表空间会自动重定向至新的路径。但是，不能通过使用 SET TABLESPACE CONTAINERS 命令显式重定向与自动存储器表空间相关联的容器；不允许执行此操作。

使用 db2ckbkp 命令的 -s 选项来显示是否对备份映像中的数据库启用自动存储器。如果启用了自动存储器，那么会显示与数据库相关联的存储路径。

对于启用了多分区自动存储器的数据库，RESTORE DATABASE 命令有一些额外的隐含意义:

1. 数据库必须在所有数据库分区上使用一组相同的存储路径。
2. 只能在目录数据库分区上使用新的存储路径发出 RESTORE 命令，在所有非目录数据库分区上执行此操作会将数据库的状态设置为 RESTORE_PENDING。

*表 43. 多分区数据库的 Restore 隐含意义*

| RESTORE DATABASE 命令 | 在数据库分区 # 上发出 | 磁盘上不存在同名的数据库 | | 磁盘上存在同名的数据库（包括框架数据库） | |
|---|---|---|---|---|---|
| | | 其他数据库分区上的结果 | 存储路径 | 其他数据库分区上的结果 | 存储路径 |
| RESTORE DATABASE TEST1 | 目录数据库分区 | 在目录数据库分区上使用存储路径从备份映像创建框架数据库。所有其他数据库分区都处于 RESTORE_PENDING 状态。 | 使用在备份映像中定义的存储路径 | 无。尚未更改存储路径，因此其他数据库分区没有变化 | 使用在备份映像中定义的存储路径 |
| | 非目录数据库分区 | 返回 SQL2542N 或 SQL2551N。如果数据库不存在，那么必须先复原目录数据库分区。 | 无 | 无。尚未更改存储路径，因此其他数据库分区没有变化 | 使用在备份映像中定义的存储路径 |
| RESTORE DATABASE TEST2 ON /path1, /path2, /path3 | 目录数据库分区 | 使用在 RESTORE 命令中指定的存储路径创建框架数据库。所有其他数据库分区都处于 RESTORE_PENDING 状态。 | /path1, /path2, /path3 | | /path1, /path2, /path3 |
| | 非目录数据库分区 | 返回 SQL1174N。如果数据库不存在，那么必须先复原目录数据库分区。不能在非目录数据库分区的 RESTORE 中指定存储路径。 | 无 | 返回 SQL1172N。不能在非目录数据库分区的 RESTORE 中指定新的存储路径。 | 无 |

## 编目数据库

当创建一个新数据库时，会在系统数据库目录文件中自动对它进行编目。还可以使用 CATALOG DATABASE 命令在系统数据库目录文件中显式地对数据库进行编目。

CATALOG DATABASE 命令允许您使用另一别名对数据库进行编目，或对先前使用 UNCATALOG DATABASE 命令删除的数据库条目进行编目。

虽然数据库是在创建数据库时自动编目的，但仍需要对数据库进行编目。对数据库进行编目时，该数据库必须存在。

缺省情况下，可使用“目录高速缓存支持”（*dir_cache*）配置参数来在内存中高速缓存目录文件，包括数据库目录。当启用目录高速缓存时，另一个应用程序对目录所作的更改（例如，使用 CATALOG DATABASE 或 UNCATALOG DATABASE 命令）可能要在重新启动应用程序之后才会生效。要刷新命令行处理器会话所使用的目录高速缓存，请发出 TERMINATE 命令。

在分区数据库中，在每个数据库分区上创建目录文件的高速缓存。

除应用程序级高速缓存外，数据库管理器级高速缓存也用于内部的数据库管理器查询。要刷新此“共享”高速缓存，请发出 db2stop 和 db2start 命令。

要使用命令行处理器来用另一别名对数据库进行编目，请使用 CATALOG DATABASE 命令。例如，以下命令行处理器命令将 PERSON1 数据库编目为 HUMANRES:

```
CATALOG DATABASE personl AS humanres
        WITH "Human Resources Database"
```

此处，系统数据库目录条目将使 HUMANRES 作为数据库别名，以便与数据库名称（PERSON1）区分。

要从客户机应用程序中对系统数据库目录中的数据库进行编目，请调用 sqlecadb API。

要使用命令行处理器在非缺省的实例上对数据库进行编目，请使用 CATALOG DATABASE 命令。在以下示例中，与数据库 B 的连接还连接至 INSTNC_C。在尝试此命令前，实例 instnc_c 必须已编目为本地节点。

```
CATALOG DATABASE b as b_on_ic AT NODE instnc_c
```

**注:** 在客户机节点上还会使用 CATALOG DATABASE 命令来对位于数据库服务器上的数据库进行编目。

## 将实用程序绑定至数据库

创建数据库时，数据库管理器尝试将 db2ubind.lst 和 db2cli.lst 中的实用程序绑定至数据库。这些文件存储在 sqllib 目录的 bnd 子目录中。

绑定一个实用程序将创建一个*程序包*，该程序包是这样一个对象，它包括处理单个源文件中特定 SQL 和 XQuery 语句所需的所有信息。

**注:** 如果希望从客户机使用这些实用程序，那么必须显式地将它们绑定。必须位于这些文件所在的目录中，才能在 sample 数据库中创建程序包。可在 sqllib 目录的 bnd 子目录中找到这些绑定文件。从客户机中创建或升级数据库时，还必须绑定 db2schema.bnd 文件。请参阅“DB2 CLI 绑定文件和程序包名称”以了解详细信息。

要将实用程序绑定或重新绑定至数据库，请在命令行中调用下列命令，其中 sample 是数据库的名称:

```
connect to sample
   bind @db2ubind.lst
```

## 创建数据库别名

*别名*是引用表、昵称或视图的间接方法，这样 SQL 或 XQuery 语句可与该表或视图的限定名无关。

仅当表名或视图名更改的情况下，才必须更改别名定义。可以在一个别名上创建另一个别名。别名可以在视图或触发器定义以及任何 SQL 或 XQuery 语句中使用，但表检查约束定义除外，在该定义中可以引用现有的表名或视图名。

可以为定义时不存在的表、视图或别名定义别名。但是，当编译包含该别名的 SQL 或 XQuery 语句时，它必须存在。

别名可以在任何可使用现有表名的地方使用，且在别名链中不存在循环引用或重复引用的情况下，可以引用另一个别名。

别名不能与现有的表、视图或别名同名，而只能引用同一个数据库中的一个表。在 CREATE TABLE 或 CREATE VIEW 语句中使用的表或视图的名称不能与相同模式中的别名相同。

除非别名所处的模式不是您当前的授权标识所拥有的模式（它需要 DBADM 权限），否则，创建别名不需要特权。

当删除一个别名或别名引用的对象时，从属于该别名的所有程序包就会标记为无效，而从属于该别名的所有视图和触发器则标记为不可用。

要使用命令行来创建别名，请输入:

```
CREATE ALIAS <alias_name> FOR <table_name>
```

在编译语句时，别名被表名或视图名替换。如果别名或别名链不能被解析为表名或视图名，那么将导致错误。例如，如果 WORKERS 是 EMPLOYEE 的一个别名，那么在编译时:

```
 SELECT * FROM WORKERS
```

就会使以下语句生效

```
SELECT * FROM EMPLOYEE
```

下列 SQL 语句为 EMPLOYEE 表创建别名 WORKERS:

```
CREATE ALIAS WORKERS FOR EMPLOYEE
```

**注**: DB2 OS/390 版或 z/Series 版使用两种不同概念的别名: ALIAS 和 SYNONYM。这两种概念在 DB2 数据库中是有区别的，如下所示:

- DB2 OS/390 版或 z/Series 版中的 ALIASe:
  - 要求它们的创建者具有特殊的权限或特权
  - 不能引用其他别名
- DB2 OS/390 版或 z/Series 版中的 SYNONYM:
  - 只能被它们的创建者使用
  - 始终是非限定的
  - 删除引用的表时被删除
  - 不要与表或视图共享名称空间

## 连接至分布式关系数据库

分布式关系数据库是在正式请求器/服务器协议和函数的基础上构建的。

*应用程序请求器*支持连接的应用程序端。它将应用程序发出的数据库请求变换为适合在分布式数据库网络中使用的通信协议。这些请求由连接的另一端中的*数据库服务器*接收和处理。应用程序请求器和数据库服务器一起处理通信和位置注意事项，以便应用程序可以像正在访问本地数据库一样操作。

应用程序进程必须先连接至数据库管理器的应用程序服务器，然后才能执行引用表或视图的 SQL 语句。CONNECT 语句在应用程序进程与其服务器之间建立连接。

有两种类型的 CONNECT 语句:

- CONNECT（1 类）支持每个工作单元（远程工作单元）一个数据库的语义。
- CONNECT（2 类）支持每个工作单元（应用程序导向的分布式工作单元）多个数据库的语义。

DB2 调用级接口（CLI）和嵌入式 SQL 支持称为*并发事务*的连接方式，该方式允许多个连接，并且每个连接都是一个独立的事务。一个应用程序可以有多个与同一数据库的并发连接。

对于启动进程的环境来说，应用程序服务器可以是本地或远程的。即使环境未在使用分布式关系数据库，应用程序服务器也存在。此环境包括一个本地目录，该目录描述可以在 CONNECT 语句中标识的应用程序服务器。

应用程序服务器运行引用表或视图的绑定形式的静态 SQL 语句。绑定语句来自数据库管理器先前通过绑定操作创建的程序包。

在大多数情况下，连接至应用程序服务器的应用程序可以使用应用程序服务器的数据库管理器支持的语句和子句。即使应用程序正在通过不支持其中某些语句和子句的数据库管理器的应用程序请求器运行也是如此。

## 分布式关系数据库的远程工作单元

*远程工作单元工具*可以远程预编译并执行 SQL 语句。

计算机系统 A 中的应用程序进程可以连接至计算机系统 B 中的应用程序服务器，并且将在一个或多个工作单元内执行引用系统 B 中的对象的任意数目静态或动态 SQL 语句。结束系统 B 中的工作单元后，该应用程序进程可以连接至计算机系统 C 中的应用程序服务器，并依此类推。

在遵循下列限制的情况下，大多数 SQL 语句可以远程预编译并执行:

- 在单个 SQL 语句中引用的所有对象必须由同一应用程序服务器管理。
- 一个工作单元中的所有 SQL 语句必须由同一应用程序服务器执行。

在任意给定时间，应用程序进程都可以处于下列四个可能的*连接状态*之一:

- 可连接并已连接

  应用程序进程已连接至应用程序服务器，并且可以执行 CONNECT 语句。

  如果隐式连接可用:

  – 从可连接但未连接状态成功执行 CONNECT TO 语句或不带任何操作数的 CONNECT 语句后，应用程序进程将进入此状态。

– 如果发出了除 CONNECT RESET、DISCONNECT、SET CONNECTION 或 RELEASE 之外的任何 SQL 语句，那么应用程序进程可以从可隐式连接状态进入此状态。

无论隐式连接是否可用，在下列情况下都将进入此状态:

– 从可连接但未连接状态成功执行了 CONNECT TO 语句。
– 从不可连接但已连接状态成功发出了 COMMIT 或 ROLLBACK 语句，或者强制执行了回滚。

• 不可连接但已连接

应用程序进程已连接至应用程序服务器，但无法成功执行 CONNECT TO 语句来更改应用程序服务器。当应用程序进程执行除下列语句外的任何 SQL 语句时，它将从可连接并已连接状态进入此状态: CONNECT TO、不带任何操作数的 CONNECT、CONNECT RESET、DISCONNECT、SET CONNECTION、RELEASE、COMMIT 或 ROLLBACK。

• 可连接但未连接

应用程序进程未连接至应用程序服务器。CONNECT TO 是唯一可以执行的 SQL 语句；否则，将产生错误（SQLSTATE 08003）。

无论隐式连接是否可用，如果在发出 CONNECT TO 语句时发生错误，或者由于工作单元内发生错误而导致连接断开并回滚，那么应用程序进程将进入此状态。如果错误是由于应用程序进程未处于可连接状态或服务器名称未列示在本地目录中而产生的，那么此错误不会导致过渡到此状态。

如果隐式连接不可用:

– 应用程序进程最初处于此状态
– CONNECT RESET 和 DISCONNECT 语句将导致过渡到此状态。

• 可隐式连接（如果隐式连接可用）。

如果隐式连接可用，那么这是应用程序进程的初始状态。CONNECT RESET 语句将导致过渡到此状态。如果在不可连接但已连接状态下发出 COMMIT 或 ROLLBACK 语句，接着在可连接并已连接状态下发出 DISCONNECT 语句，那么也会导致进入此状态。

隐式连接的可用性由安装选项、环境变量和认证设置确定。

由于 CONNECT 语句本身不会使应用程序进程脱离可连接状态，所以连续执行 CONNECT 语句不是错误的行为。但是，连续执行 CONNECT RESET 语句却是错误行为。先执行除下列语句外的任何 SQL 语句，然后执行 CONNECT TO 语句也是错误行为: CONNECT TO、CONNECT RESET、不带任何操作数的 CONNECT、SET CONNECTION、RELEASE、COMMIT 或 ROLLBACK。为了避免这种错误，在执行 CONNECT TO 语句之前，应执行 CONNECT RESET、DISCONNECT（前面是 COMMIT 或 ROLLBACK 语句）、COMMIT 或 ROLLBACK 语句。

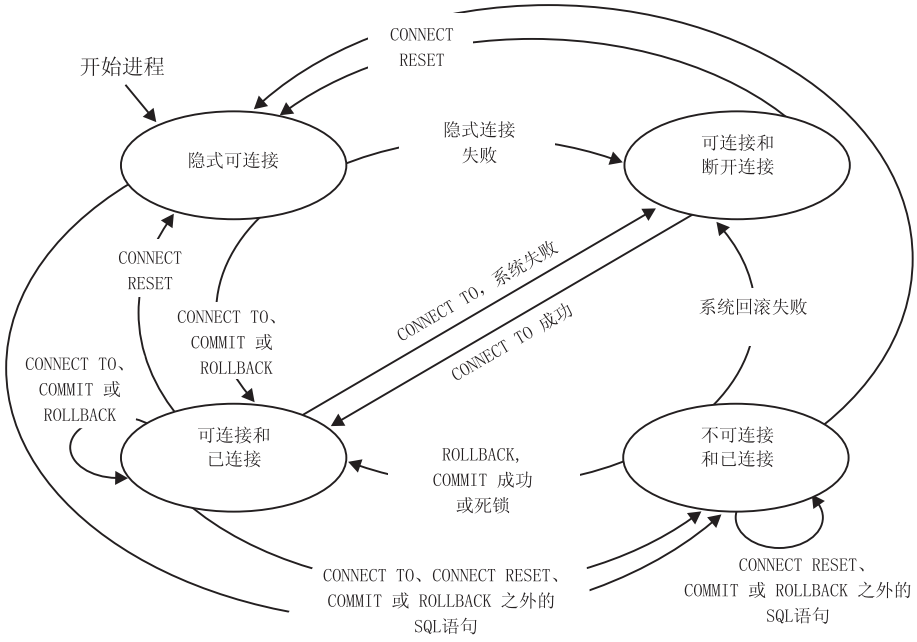



*图 3. 隐式连接可用时的连接状态过渡*

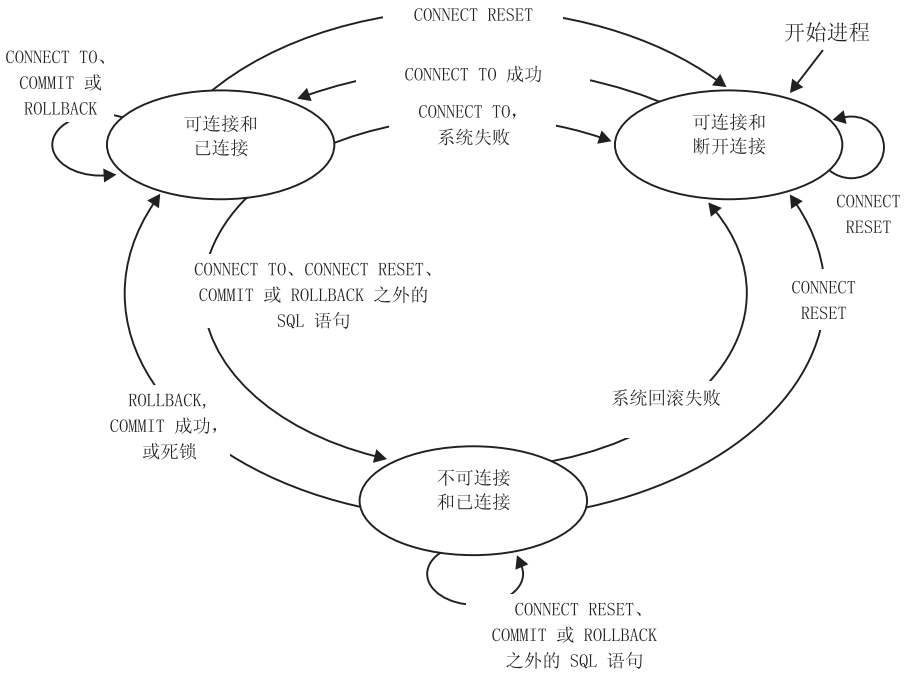



*图 4. 隐式连接不可用时的连接状态过渡*

## 应用程序导向的分布式工作单元

*应用程序导向的分布式工作单元工具*可以远程预编译并执行 SQL 语句。

通过发出 CONNECT 或 SET CONNECTION 语句，计算机系统 A 中的应用程序进程可以连接至计算机系统 B 中的应用程序服务器。然后，在结束工作单元之前，该应用程序进程可以执行引用系统 B 中的对象的任意数目静态和动态 SQL 语句。在单个 SQL 语句中引用的所有对象必须由同一应用程序服务器管理。但是，与远程工作单元工具不同，任意数目的应用程序服务器可以参与同一工作单元。落实或回滚操作将结束工作单元。

应用程序导向的分布式工作单元使用 2 类连接。*2 类*连接将应用程序进程连接至已识别的应用程序服务器，并为应用程序导向的分布式工作单元制订规则。

2 类应用程序进程:

- 始终可连接
- 处于已连接状态或未连接状态
- 没有连接或有多个连接。

应用程序进程的每个连接都由连接的应用程序服务器的数据库别名唯一标识。

单个连接始终具有下列其中一种连接状态:

- 当前和挂起
- 当前和释放暂挂
- 休止和挂起
- 休止和释放暂挂

2 类应用程序进程最初处于未连接状态，并且没有任何连接。连接最初处于当前和挂起状态。

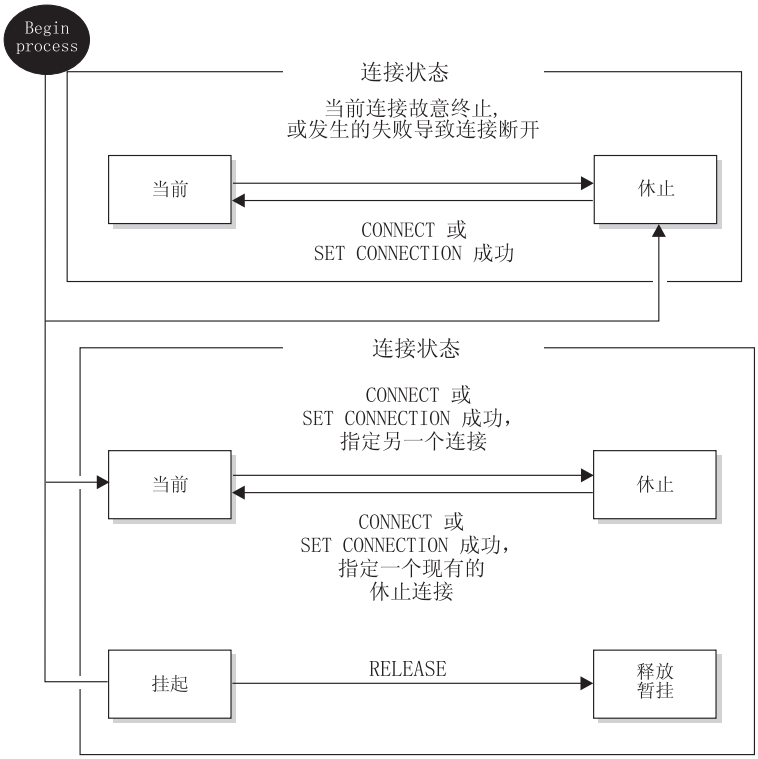



*图 5. 应用程序导向的分布式工作单元状态过渡*

## 应用程序进程连接状态

执行 CONNECT 语句时要遵循某些规则。

执行 CONNECT 语句时要遵循下列规则:

- 一个上下文不能同时与同一应用程序服务器具有多个连接。
- 当应用程序进程执行 SET CONNECTION 语句时，指定的位置名必须是该应用程序进程的连接集中的现有连接。
- 当应用程序进程执行 CONNECT 语句并且 SQLRULES(STD) 选项有效时，指定的服务器名称不能是该应用程序进程的连接集中的现有连接。有关 SQLRULES 选项的描述，请参阅第 126 页的『控制工作单元语义的选项』。

**如果应用程序进程具有当前连接**，那么该应用程序进程处于*已连接*状态。CURRENT SERVER 专用寄存器包含当前连接的应用程序服务器名称。应用程序进程可以执行引用该应用程序服务器所管理的对象的 SQL 语句。

当处于未连接状态的应用程序进程成功执行 CONNECT 或 SET CONNECTION 语句后，它将进入已连接状态。如果没有连接，但发出了 SQL 语句，那么只要使用缺省数据库名称设置了 DB2DBDFT 环境变量，就会建立隐式连接。

**如果应用程序进程没有当前连接**，那么该应用程序进程处于*未连接*状态。唯一可以执行的 SQL 语句有 CONNECT、DISCONNECT ALL、DISCONNECT（指定数据库）、SET CONNECTION、RELEASE、COMMIT、ROLLBACK 和本地 SET 语句。

如果处于*已连接状态*的应用程序进程的当前连接有意结束，或者由于 SQL 语句失败而导致应用程序服务器中执行回滚操作且连接中断，那么该应用程序进程将进入*未连接状态*。成功执行 DISCONNECT 语句，或者在连接处于释放暂挂状态时成功执行 COMMIT 语句后，连接会有意结束。（如果 DISCONNECT 预编译程序选项设置为 AUTOMATIC，那么所有连接都将结束。如果它设置为 CONDITIONAL，那么没有打开的 WITH HOLD 游标的所有连接都将结束。）

## 连接状态

有两种类型的连接状态: “挂起和释放暂挂状态”以及“当前和休止状态”。

如果应用程序进程执行 CONNECT 语句并且应用程序请求器知道服务器名称，但该名称不在该应用程序进程的现有连接集中，那么: (i) 将使当前连接处于*休止连接状态*、该服务器名称将添加到连接集中，并且将使新连接同时处于*当前连接状态*和*挂起连接状态*。

如果服务器名称已经位于应用程序进程的现有连接集中，并且使用 SQLRULES(STD) 选项预编译了应用程序，那么会产生错误（SQLSTATE 08002）。

***挂起和释放暂挂状态。***RELEASE 语句控制连接处于挂起还是释放暂挂状态。*释放暂挂*状态意味着在下一个成功落实操作时将断开连接。（回滚对连接不起作用。）*挂起*状态意味着在下一个落实操作时*不*会断开连接。

所有连接最初都处于挂起状态，并且可以使用 RELEASE 语句转至释放暂挂状态。连接处于释放暂挂状态后，就不能将它恢复为挂起状态。如果发出了 ROLLBACK 语句，或者由于落实操作不成功而导致执行回滚操作，那么连接将在工作单元边界保持释放暂挂状态。

即使某个连接未显式标记为释放，如果落实操作符合 DISCONNECT 预编译器选项的条件，那么该连接仍可能被落实操作断开。

***当前和休止状态。***无论连接处于挂起状态还是释放暂挂状态，它还可以处于当前状态或休止状态。在连接处于*当前*状态期间，该连接就是用来执行 SQL 语句的连接。处于*休止*状态的连接不是当前连接。

唯一可以在休止连接上流动的 SQL 语句是 COMMIT、ROLLBACK、DISCONNECT 或 RELEASE。SET CONNECTION 和 CONNECT 语句将指定服务器的连接状态更改为当前，并且将使任何现有连接处于或保持休止状态。在任何时间点，只能有一个连接处于当前状态。如果休止连接成为同一工作单元中的当前连接，那么所有锁定、游标和预编译语句的状态与该连接上次为当前连接时它们所处的状态相同。

### 连接结束时

连接结束时，应用程序进程通过该连接获取的所有资源以及用来创建和维护该连接的所有资源都取消分配。例如，如果应用程序进程执行 RELEASE 语句，那么当连接在下一个落实操作期间结束时，所有打开的游标都关闭。

连接还可能由于通信故障而结束。如果此连接处于当前状态，那么应用程序进程将处于未连接状态。

当应用程序进程结束时，该进程的所有连接都将结束。

## 控制工作单元语义的选项

2 类连接管理的语义由一组预编译程序选项确定。下面概述了这些选项，其中缺省值用粗体和加下划线文本表示。

- CONNECT (**1** | 2)。指定将 CONNECT 语句作为 1 类还是 2 类处理。
- SQLRULES (**DB2** | STD)。指定是根据 DB2 规则还是 SQL92 标准规则来处理 2 类 CONNECT，其中 DB2 规则允许 CONNECT 切换至休止连接，而 SQL92 标准规则不允许这样做。
- DISCONNECT (**EXPLICIT** | CONDITIONAL | AUTOMATIC)。指定在执行落实操作时要断开连接的数据库连接:
  - 显式标记为要由 SQL RELEASE 语句释放的数据库连接（EXPLICIT）
  - 没有打开的 WITH HOLD 游标的数据库连接以及标记为释放的数据库连接（CONDITIONAL）
  - 所有连接（AUTOMATIC）。
- SYNCPOINT (**ONEPHASE** | TWOPHASE | NONE)。指定如何在多个数据库连接之间协调 COMMIT 或 ROLLBACK。将忽略此选项，包括它只是为了提供向后兼容性。
  - 只能对工作单元中的一个数据库进行更新，所有其他数据库都是只读数据库（ONEPHASE）。对其他数据库进行的任何更新尝试都会导致错误（SQLSTATE 25000）。
  - 在运行时使用事务管理器（TM）来协调支持此协议的那些数据库之间的两阶段落实（TWOPHASE）。
  - 不要使用 TM 来执行两阶段落实，并且不要强制执行单个更新程序和多个阅读器（NONE）。执行 COMMIT 或 ROLLBACK 语句时，将各个 COMMIT 或 ROLLBACK 记入所有数据库。如果一个或多个 ROLLBACK 失败，那么会产生错误（SQLSTATE 58005）。如果一个或多个 COMMIT 失败，那么会产生另一个错误（SQLSTATE 40003）。

要在运行时覆盖上述任何选项，请使用 SET CLIENT 命令或 sqlesetc 应用程序编程接口（API）。可以使用 QUERY CLIENT 命令或 sqleqryc API 来获取 SET CLIENT 或 sqlesetc 的当前设置。请注意，QUERY CLIENT 或 sqleqryc API 不是 SQL 语句，它们是在各种主语言和命令行处理器（CLP）中定义的 API。

## 数据表示注意事项

不同的系统使用不同方式来表示数据。将数据从一个系统移至另一个系统时，有时必须执行数据转换。

支持 DRDA 的产品将自动在接收系统中执行任何必需的转换。

要对数字数据执行转换，系统需要知道数据类型以及发送系统中表示该数据类型的方式。转换字符串需要其他信息。字符串转换取决于数据的代码页和要对该数据执行的操作。根据 IBM 字符数据表示体系结构（CDRA）执行字符转换。有关字符转换的更多信息，请参阅 *Character Data Representation Architecture: Reference & Registry*（SC09-2190-00）手册。

## 查看本地或系统数据库目录文件

使用 LIST DATABASE DIRECTORY 命令来查看与系统上的数据库相关联的信息。

在查看本地或系统数据库目录文件之前，必须先创建实例和数据库。

要查看本地数据库目录文件的内容，请发出以下命令，其中 <location> 指定该数据库的位置:

```
LIST DATABASE DIRECTORY ON <location>
```

要查看系统数据库目录文件的内容，请发出 LIST DATABASE DIRECTORY 命令而不指定该数据库目录文件的位置。

## 删除数据库

因为删除数据库会删除它的所有对象、容器和相关的文件，所以此操作可产生广泛的影响。删除的数据库将从数据库目录中除去（取消编目）。

要使用命令行来删除数据库，请输入:

```
DROP DATABASE <name>
```

以下命令删除数据库 SAMPLE:

```
DROP DATABASE SAMPLE
```

**注**: 如果已删除 SAMPLE 数据库而又发现再次需要它，那么可重新创建。

要从客户机应用程序中删除数据库，请调用 sqledrpd API。要在指定的数据库分区服务器上删除数据库，请调用 sqledpan API。

### 删除别名

删除别名时，将从目录中删除其描述、使引用该别名的任何程序包和高速缓存的动态查询失效，并且将从属于该别名的所有视图和触发器都标记为不可用。

要删除别名，请在命令行中发出以下 DROP 语句:

```
DROP ALIAS EMPLOYEE-ALIAS
```

# 第 7 章 数据库分区

*数据库分区*是数据库的一部分，它由其自己的数据、索引、配置文件和事务日志组成。数据库分区有时称为节点或数据库节点。分区数据库环境是支持将数据分发到各数据库分区上的数据库安装。

有关数据库分区的完整详细信息，请参阅《*分区和集群指南*》。

© Copyright IBM Corp. 1993, 2010

# 第 8 章 缓冲池

*缓冲池*指的是从磁盘读取表和索引数据时，数据库管理器分配的用于高速缓存这些表或索引数据的主存储器区域。每个 DB2 数据库都必须具有一个缓冲池。

每个新数据库都定义了一个称为 IBMDEFAULTBP 的缺省缓冲池。可以使用 CREATE BUFFERPOOL、DROP BUFFERPOOL 和 ALTER BUFFERPOOL 语句来创建、删除和修改缓冲池。SYSCAT.BUFFERPOOLS 目录视图访问数据库中所定义的缓冲池的信息。

### 缓冲池的使用方法

首次访问表中的数据行时，数据库管理器会将包含该数据的页放入缓冲池中。这些页将一直保留在缓冲池中，直到关闭数据库或者其他页需要使用某一页所占用的空间为止。

缓冲池中的页可能正在使用，也可能没有使用，它们可能是脏页，也可能是干净页:

- 正在使用的页就是当前正在读取或更新的页。为了保持数据一致性，数据库管理器只允许一次只有一个代理程序更新缓冲池中的给定页。如果正在更新某页，那么它正在内一个代理程序互斥地访问。如果正在读取该页，那么多个代理程序可以同时读取该页。
- “脏”页包含已更改但尚未写入磁盘的数据。
- 将一个已更改的页写入磁盘之后，它就是一个“干净”页，并且可能仍然保留在缓冲池中。

大多数情况下，调整数据库涉及到设置用于控制将数据移入缓冲池以及等待将数据从缓冲区写入磁盘的配置参数。如果最近的代理程序不需要页空间，那么可以将页空间用于新应用程序中的新页请求。额外的磁盘 I/O 会使数据库管理器性能下降。

可使用快照监视器来计算缓冲池命中率，缓冲池命中率可帮助您调整缓冲池。

## 设计缓冲池

所有缓冲池的大小可能对数据库性能产生重要影响。

在创建新的缓冲池之前，应解决下列各项:

- 想要使用什么缓冲池名称？
- 是立即创建缓冲池，还是在下一次取消激活然后重新激活数据库之后创建缓冲池？
- 是所有数据库分区都应存在缓冲池，还是一部分数据库分区应存在缓冲池？
- 希望缓冲池的页大小是多大？请参阅下面的第 132 页的『缓冲池页大小』。
- 是缓冲池将为固定大小，还是数据库管理器将自动调整缓冲池大小以对工作负载作出响应？建议您在创建缓冲池期间不指定 SIZE 参数，从而允许数据库管理器自动调整缓冲池。有关详细信息，请参阅“CREATE BUFFERPOOL 语句”的 SIZE 参数以及第 132 页的『缓冲池内存注意事项』。
- 您是否想保留一部分缓冲池用于基于块的 I/O？有关详细信息，请参阅“经过改进的顺序预取的基于块的缓冲池”。

© Copyright IBM Corp. 1993, 2010

## 表空间与缓冲池之间的关系

设计缓冲池时，需要了解表空间与缓冲池之间的关系。每个表空间都与一个特定的缓冲池相关。IBMDEFAULTBP 是缺省缓冲池。数据库管理器还会分配下列系统缓冲池: IBMSYSTEMBP4K、IBMSYSTEMBP8K、IBMSYSTEMBP16K 和 IBMSYSTEMBP32K（以前称为“隐藏缓冲池”）。要使另一个缓冲池与表空间相关，那么该缓冲池必须存在并且它们具有相同的页大小。关联是在使用 CREATE TABLESPACE 语句创建表空间时定义的，但以后可使用 ALTER TABLESPACE 语句更改此关联。

如果拥有多个缓冲池，那么可以配置数据库使用的内存，以改善整体性能。例如，如果具有带有一个或多个用户可随机访问的大（大于可用内存）表的表空间，缓冲池的大小可能受到限制，因为高速缓存该数据页可能没有好处。用于联机事务应用程序的表空间可以与一个较大的缓冲池相关联，以便可以更长时间地高速缓存应用程序所使用的数据页，导致响应时间更快。配置新缓冲池时必须小心。

## 缓冲池页大小

缺省缓冲池的页大小是在使用 CREATE DATABASE 命令时设置的。此缺省值表示所有将来 CREATE BUFFERPOOL 和 CREATE TABLESPACE 语句的缺省页大小。如果在创建数据库时不指定页大小，那么缺省页大小是 4 KB。

**注:** 如果确定数据库需要 8 KB、16 KB 或 32 KB 的页大小，那么必须至少定义一个具有相匹配的页大小并且与数据库中的表空间相关联的缓冲池。

但是，您可能需要具有与系统缓冲池不同的特征的缓冲池。可以为要使用的数据库管理器创建新的缓冲池。可能必须重新启动数据库，才能使表空间和缓冲池更改生效。为表空间指定的页大小应确定为缓冲池选择的页大小。选择用于缓冲池的页大小是很重要的，这是因为创建缓冲池之后就不能改变页大小了。

## 缓冲池内存注意事项

### 内存需求

设计缓冲池时，还应根据机器上已安装的内存量以及与数据库管理器在同一机器上同时运行的其他应用程序所需要的内存来考虑内存需求。当没有足够内存来保存所访问的所有数据时，操作系统就会进行数据交换。将某些数据写入或交换到临时磁盘存储器中以对其他数据腾出空间时就会进行数据交换。当需要临时磁盘存储器上的数据时，又会将数据交换回到主存储器中。

### 缓冲池内存保护

对于版本 9.5，缓冲池内存中的数据页是使用存储密钥保护的，仅当在 POWER6® 上运行的 AIX（5.3 TL06 5.4）上由 DB2_MEMORY_PROTECT 注册表变量显式启用时，存储密钥才可用。

缓冲池内存保护在每个代理程序级别工作；当任何特定代理程序需要访问时，该代理程序仅访问缓冲池页面。内存保护通过确定 DB2 引擎线程应访问缓冲池内存的时间来工作。有关详细信息，请参阅: 第 133 页的『缓冲池内存保护（在 POWER6 上运行的 AIX）』。

### 地址窗口扩展（AWE）和扩充存储器（ESTORE）

**注:** 已经废止了 AWE 和 ESTORE 功能，包括与 ESTORE 相关的关键字、监视元素和数据结构。要分配更多内存，需要升级到 64 位硬件操作系统和相关联的 DB2 产品。还应该修改应用程序和脚本，以除去对此已废止的功能的引用。

## 缓冲池内存保护（在 POWER6 上运行的 AIX）

数据库管理器使用缓冲池来对大量数据库数据应用添加、修改和删除。在 POWER6 上运行的 AIX 5.3 TL06+ 上，可以使用*存储密钥*来保护缓冲池内存。

*存储密钥*是 IBM® Power6 处理器和 AIX® 操作系统中的一项新功能，它允许在内核线程级别使用硬件密钥来保护某些范围的内存。存储密钥保护将减少缓冲池内存被毁坏的问题，还会限制可能会使数据库停止的错误数。尝试通过编程方法非法访问缓冲池将导致错误情况，数据库管理器可以检测到此错误情况并进行处理。

**注:** 缓冲池内存保护在每个代理程序级别工作；当任何特定代理程序需要访问时，该代理程序仅访问缓冲池页面。

数据库管理器通过限制访问缓冲池内存来保护缓冲池。当代理程序需要访问缓冲池以执行工作时，将临时授权它访问缓冲池内存。当代理程序不再需要访问缓冲池时，就会撤销访问。这样可确保只有绝对需要时才允许代理程序修改缓冲池的内容，从而降低缓冲池被毁坏的可能性。非法访问缓冲池内存都会导致分段错误。提供了一些工具来诊断这些错误，例如 db2diag、db2fodc、db2pdcfg 和 db2support 命令。

要完全启用此缓冲池内存保护功能，为了增大数据库引擎的弹性，应同时启用 DB2_MEMORY_PROTECT 和 DB2_THREAD_SUSPENSION 注册表变量:

**DB2_MEMORY_PROTECT 注册表变量**

此注册表变量可启用和禁用缓冲池内存保护功能。当启用 DB2_MEMORY_PROTECT（设置为 YES）时，如果 DB2 引擎线程试图非法访问缓冲池内存，那么该引擎线程就会被捕获。缺省值为 NO。

**DB2_THREAD_SUSPENSION 注册表变量**

此注册表变量可启用和禁用 DB2 线程暂挂功能。它允许您通过暂挂出现问题的引擎线程（即，尝试非法访问受存储密钥保护的内存的线程）来控制 DB2 实例是否仍然落入陷阱。.

**注意:** 缓冲池内存保护功能取决于 AIX Storage Protect Keys（存储保护键）的实现，并且此功能对于固定共享内存可能无法正常使用。如果通过 DB2_PINNED_BP 或 DB2_LARGE_PAGE_MEM 设置指定了 DB2_MEMORY_PROTECT，那么可能无法启用 AIX Storage Protect Keys（存储保护键）。有关 AIX Storage Protect Keys（存储保护键）的更多信息，请参阅以下链接: http://publib.boulder.ibm.com/infocenter/systems/scope/aix/index.jsp?topic=/com.ibm.aix.genprogc/doc/genprogc/storage_protect_keys.htm

仅当启用了 DB2_MEMORY_PROTECT 时，才能启用 DB2_THREAD_SUSPENSION。如果启用了 DB2_THREAD_SUSPENSION:

- 无论线程中发生了哪种故障，不管此故障是否是由于尝试访问使用存储密钥保护并且该线程对其没有访问权的内存而导致的，如果是在线程对使用存储密钥保护的此内存没有访问权的位置发生了此故障，那么数据库管理器会保证受保护的内存未被毁坏，从而允许 DB2 引擎继续运行。
- 当以不受防护方式运行不带 SQL 的用户定义的函数时，如果检测到缓冲池内存保护违例，那么数据库管理器会对该 UDF 的调用者返回错误，并且数据库将继续运行而不受影响。

如果 DB2_LGPAGE_BP 设置为 YES，那么您将无法使用内存保护。即使 DB2_MEMORY_PROTECT 设置为 YES，DB2 也无法保护缓冲池内存并禁用此功能部件。

## 创建缓冲池

使用 CREATE BUFFERPOOL 语句来定义数据库管理器要使用的新缓冲池。

以下是基本 CREATE BUFFERPOOL 语句的一个示例:

```
CREATE BUFFERPOOL <buffer pool name>
  PAGESIZE 4096
```

如果有足够的内存可用，那么缓冲池可能会立即变为活动状态。缺省情况下，新的缓冲池是使用 IMMEDIATE 关键字创建的，在大多数平台上，数据库管理器将能够获得更多内存。期望返回的是成功分配了内存。只有在数据库管理器无法分配更多内存的情况下才会返回警告情况，指出未能启动缓冲池，将在下一次启动数据库时再启动缓冲池。对于立即请求，不需要重新启动数据库。落实此语句时，缓冲池将反映在系统目录表中，但缓冲池要在下次启动数据库后才变成活动状态。有关此语句（包括其他选项）的更多信息，请参阅“CREATE BUFFERPOOL 语句”。

如果您发出 CREATE BUFFERPOOL DEFERRED，那么不会立即激活缓冲池；将在下一次启动数据库时创建缓冲池。在重新启动数据库之前，任何新的表空间都将使用现有缓冲池，即使创建该表空间时规定它显式使用延迟缓冲池也是如此。

机器上需要有足够的实内存用于已创建的所有缓冲池。操作系统还需要一些内存才能运行。

要使用命令行来创建缓冲池，请执行以下操作:

1. 通过发出以下 SQL 语句来获取数据库中已存在的缓冲池名称的列表:

   ```
   SELECT BPNAME FROM SYSCAT.BUFFERPOOLS
   ```

2. 选择当前在结果列表中找不到的缓冲池名称。
3. 确定将创建的缓冲池的特征。
4. 确保您具有正确的授权标识来运行 CREATE BUFFERPOOL 语句。
5. 发出 CREATE BUFFERPOOL 语句。

在分区数据库上，还可以定义要在每个数据库分区上以不同方式创建（包括不同的大小）的缓冲池。缺省 ALL DBPARTITIONNUMS 子句指示将在数据库中的所有数据库分区上创建此缓冲池。

在以下示例中，可选的 DATABASE PARTITION GROUP 子句标识缓冲池定义适用于的数据库分区组:

```
CREATE BUFFERPOOL <buffer pool name>
  PAGESIZE 4096
  DATABASE PARTITION GROUP <db partition group name>
```

如果指定了此参数，那么仅在这些数据库分区组中的数据库分区上创建缓冲池。每个数据库分区组当前必须存在于数据库中。如果未指定 DATABASE PARTITION GROUP 子句，那么将在所有数据库分区上（以及后来添加至数据库的任何数据库分区上）创建此缓冲池。

有关更多信息，请参阅“CREATE BUFFERPOOL 语句”。

## 修改缓冲池

有许多理由要修改缓冲池，例如，为了启用自调整内存功能。为此，可以使用 ALTER BUFFERPOOL 语句。

语句的授权标识必须具有 SYSCTRL 或 SYSADM 权限。

使用缓冲池时，可能需要完成下列其中一项任务:

- 启用缓冲池自调整功能，从而允许数据库管理器根据工作负载调整缓冲池大小。
- 修改基于块的 I/O 的缓冲池的块区域。
- 将此缓冲池定义添加到新的数据库分区组中。
- 修改部分或所有数据库分区上的缓冲池大小。

要使用命令行来改变缓冲池，请执行以下操作:

1. 要获取数据库中已存在的缓冲池名称的列表，请发出以下语句:

   ```
   SELECT BPNAME FROM SYSCAT.BUFFERPOOLS
   ```

2. 从结果列表中选择缓冲池名称。
3. 确定需要进行的更改。
4. 确保您具有正确的授权标识来运行 ALTER BUFFERPOOL 语句。

   **注**: 两个关键参数是 IMMEDIATE 和 DEFERRED。当使用 IMMEDIATE 参数时，将立即更改缓冲池大小，而不必等到下一次激活数据库时才生效。如果数据库共享内存不足以分配新空间，那么会延迟（DEFERRED）运行该语句。

   当使用 DEFERRED 参数时，要在重新激活数据库后才应用对缓冲池所作的更改。不需要保留内存空间；数据库管理器在激活时从系统中分配必需的内存。

5. 使用 ALTER BUFFERPOOL 语句来改变缓冲池对象的单个属性。例如:

   ```
   ALTER BUFFERPOOL buffer pool name SIZE number of pages
   ```

- *buffer pool name* 是由一部分组成的名称，它标识系统目录中描述的缓冲池。
- *number of pages* 是要分配给此特定缓冲池的新页数。也可以使用值 -1，它指示缓冲池大小应该是在 **buffpage** 数据库配置参数中找到的值。

该语句还可以具有 DBPARTITIONNUM <db partition number> 子句，它指定将在其上修改缓冲池大小的数据库分区。如果未指定此子句，那么将在所有数据库分区（但在 SYSCAT.BUFFERPOOLDBPARTITIONS 中具有异常条目的那些数据库分区除外）上修改缓冲池大小。有关将此子句用于数据库分区的详细信息，请参阅 ALTER BUFFERPOOL 语句。

在落实此语句时，由于此语句而对缓冲池所作的更改将反映在系统目录表中。但是，要在下次启动数据库之后，对实际缓冲池所作的更改才会生效，但是所指定的带有缺省 IMMEDIATE 关键字并成功执行的 ALTER BUFFERPOOL 请求除外。

机器上必须有足够的实内存用于创建的所有缓冲池。还需要有足够的实内存用于其余数据库管理器和应用程序。

## 删除缓冲池

删除缓冲池时，应确保没有任何表空间已指定给这些缓冲池。不能删除 IBMDEFAULTBP 缓冲池。

在下次连接至数据库之前可能不会释放磁盘存储器。在删除数据库之前，已删除的缓冲池不会真正释放存储器内存。将立即释放缓冲池内存以供数据库管理器使用。

可以使用 DROP BUFFERPOOL 语句来删除缓冲池，如下所示:

```
DROP BUFFERPOOL <buffer pool name>
```

# 第 9 章 表空间

*表空间*是一种存储结构，它包含表、索引、大对象和长型数据。表空间位于数据库分区组中。它们允许将数据库和表数据的位置直接指定到容器上。（容器可以是目录名、设备名或文件名。）这可以提供改善的性能和更灵活的配置。

因为表空间位于数据库分区组中，所以选择保存表的表空间定义了如何将该表的数据分布至数据库分区组中的数据库分区。单个表空间可跨多个容器。在同一个物理磁盘（或驱动器）上创建多个容器（从一个或多个表空间）是可能的。如果您正在使用自动存储器表空间，那么这是由数据库管理器处理的。如果未使用自动存储器表空间，为了提高性能，每个容器应使用不同的磁盘。

图 6 举例说明了一个数据库内的表和表空间与该数据库相关的容器之间的关系。

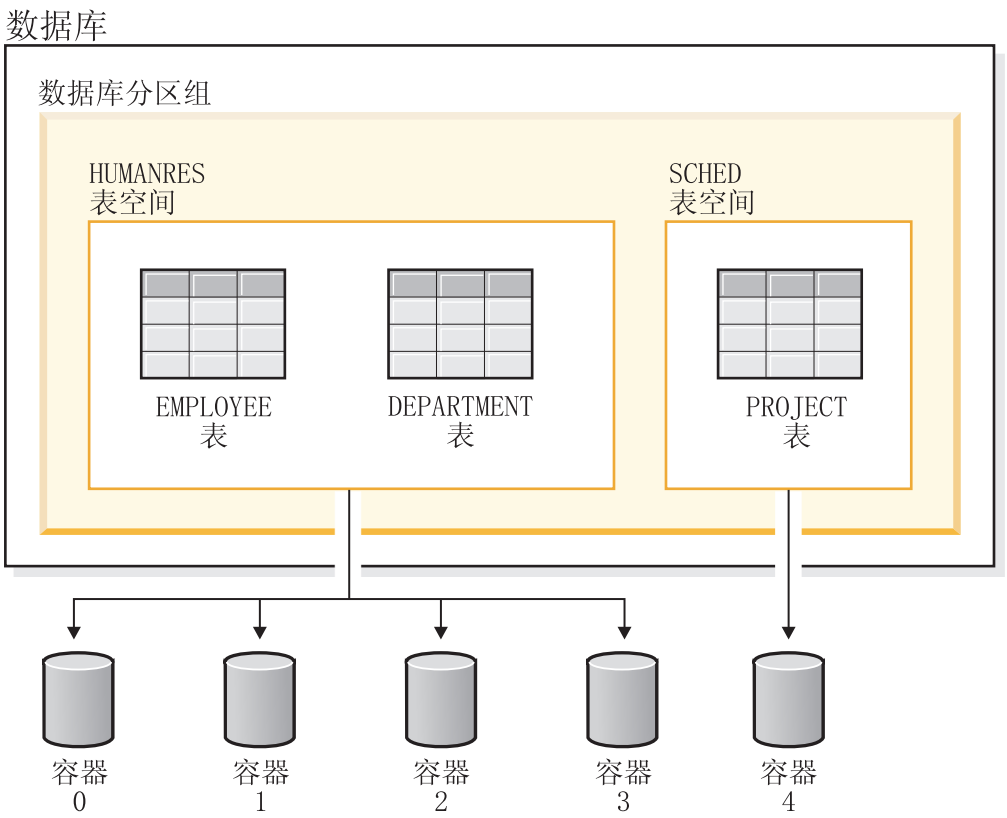



*图 6. 数据库中的表空间和表*

EMPLOYEE 和 DEPARTMENT 表在 HUMANRES 表空间中，该表空间横跨容器 0、1、2 和 3。PROJECT 表位于容器 4 中的 SCHED 表空间内。此示例显示每个容器存在于单独的磁盘中。

数据库管理器会尝试平衡分布在所有容器中的数据负荷。因此，所有容器都将用于存储数据。数据库管理器在使用另一个容器之前写入一个容器的页数称为*扩展数据块大小*。数据库管理器并非始终从第一个容器开始存储表数据。

© Copyright IBM Corp. 1993, 2010

图 7 显示具有两个 4KB 页扩展数据块大小的 HUMANRES 表空间，它有四个容器，每个容器有少量已分配的扩展数据块。DEPARTMENT 和 EMPLOYEE 表都有 7 页，且跨所有四个容器。

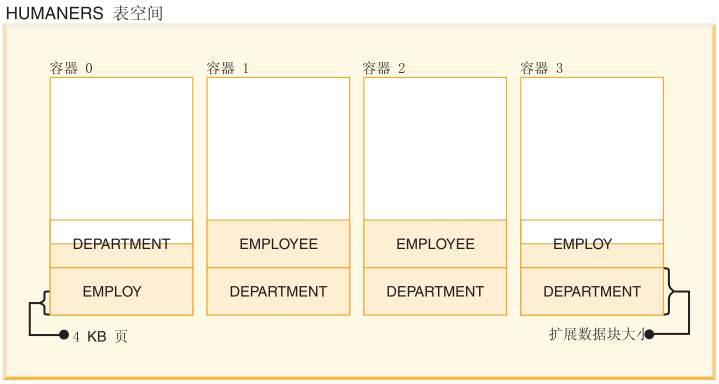



*图 7. 表空间中的容器和扩展数据块*

## 设计表空间

表空间用来指定数据库中的数据在系统上的物理存储位置，并在数据库与实际数据所在的容器对象之间提供一个间接层。

有许多理由要创建表空间，包括可恢复性和隔离不同缓冲池中的对象的能力。借助自动存储器，您不再需要关心表空间的物理磁盘位置，也不再需要关心容器的物理位置。数据库管理器自动为表空间指定或创建容器。

一个数据库至少必须包含三个表空间:

- 一个*目录表空间*，它包含该数据库的所有系统目录表。此表空间称为 SYSCATSPACE，它不能被删除。IBMCATGROUP 是此表空间的缺省数据库分区组。
- 一个或多个*用户表空间*，它们包含所有用户定义的表。缺省情况下，会创建一个表空间 USERSPACE1。IBMDEFAULTGROUP 是此表空间的缺省数据库分区组。

  当创建一个表时，应指定表空间名，否则，结果可能不是您所期望的。

  表的页大小或者由行大小确定，或者由列数确定。一行中允许的最大长度取决于创建此表所在的表空间的页大小。可能的页大小的值为 4 KB、8 KB、16 KB 和 32 KB。在版本 9.1 之前，缺省页大小为 4 KB。在版本 9.1 及其后续版本中，缺省页大小可以是其他受支持的值中的一个。缺省页大小是在创建新的数据库时声明的。声明了缺省页大小之后，仍然可以使用具有一种页大小的表空间作为表，而使用具有另一种页大小的另一个表空间来存储长型数据或 LOB 数据。（记住，SMS 不支持跨表空间的表，而 DMS 却支持。）如果列数或行大小超过表空间页大小的限制，那么返回一个错误（SQLSTATE 42997）。
- 一个或多个*临时表空间*，它们包含临时表。临时表空间可以是*系统临时表空间*或*用户临时表空间*。

系统临时表空间存放数据库管理器在执行诸如排序或连接之类的操作时所需的临时数据。这些类型的操作需要额外的空间来处理结果集。数据库必须有至少一个系统临时表空间；在缺省情况下，创建数据库时会创建一个名为 TEMPSPACE1 的系统临时表空间。IBMTEMPGROUP 是此表空间的缺省数据库分区组。

用户临时表空间存放使用 DECLARE GLOBAL TEMPORARY TABLE 语句创建的表的临时数据。为了能够定义已声明临时表，至少一个用户临时表空间应该是使用相应 USE 特权创建的。USE 特权是使用 GRANT 语句授予的。用户临时表空间*不*是在创建数据库时缺省创建的。

如果数据库使用多个临时表空间，并且需要新的临时对象，那么优化器将为此对象选择相应的页大小。然后将把该对象分配到具有相应页大小的临时表空间中。如果存在多个临时表空间具有该页大小，那么将以循环方式选择表空间。在大多数情况下，建议对于任何一个页大小，不要使用多个临时表空间。

如果对用大于缺省值的页大小定义的表空间中的表运行查询，那么某些查询可能会失败。如果没有使用更大页大小定义的临时表空间，那么会发生这种情况。可能需要创建具有更大页大小的临时表空间（如果缺省值是 4 KB，那么需要创建页大小为 8 KB、16 KB 或 32 KB 的临时表空间）。如果没有与用户表空间中的最大页大小相同的临时表空间，任何 DML（数据操作语言）语句都可能失败。

应定义一个 SMS 临时表空间，使它的页大小等于大多数用户表空间所使用的页大小。这对于典型的环境和工作负载应已足够。

### 表空间和数据库分区组

在分区数据库环境中，每个表空间都与特定数据库分区组相关。这允许将表空间的特征应用于该数据库分区组中的每个数据库分区。

该数据库分区组必须存在（用 CREATE DATABASE PARTITION GROUP 语句定义它），且当使用 CREATE TABLESPACE 语句创建表空间时定义表空间与该数据库分区组之间的关联。

不能使用 ALTER TABLESPACE 语句来更改表空间与数据库分区组之间的关联。只能为数据库分区组内的个别数据库分区更改表空间规范。在单分区环境中，每个表空间都与缺省数据库分区组相关。在定义表空间时，缺省数据库分区组为 IBMDEFAULTGROUP，除非在定义系统临时表空间，那时将使用 IBMTEMPGROUP。

## 表空间的类型

一个数据库必须至少包含三种类型的表空间：一个目录表空间、一个或多个用户表空间以及一个或多个临时表空间。

有两种类型的表空间，它们都可以在单个数据库中使用：

- 系统管理的空间，操作系统的文件管理器控制其中的存储空间。
- 数据库管理的空间，数据库管理器控制其中的存储空间。

在分区数据库环境中，目录分区将包含全部三个缺省表空间，而其他每个数据库分区将只包含 TEMPSPACE1 和 USERSPACE1。

还可以创建自动存储器表空间，该表空间将使用 SMS 或 DMS 作为基本表空间类型。数据库管理器将根据其中包含的数据类型选择实际类型 SMS 或 DMS（SMS 用于临时表空间，DMS 则用于其他表空间）。

## 系统管理的空间

在 SMS（系统管理的空间）表空间中，操作系统的文件系统管理器分配和管理用于存储表的空间。

该存储模型通常由存储在文件系统空间中的多个文件组成，这些文件表示表对象。由您决定文件的位置，数据库管理器控制它们的名称，而文件系统负责管理这些文件。通过控制写入每个文件的数据量，数据库管理器均匀地将数据分布到所有表空间容器中。

每个表至少有一个与它相关的 SMS 物理文件。

表空间中的数据按系统中所有容器上的扩展数据块进行条带分割。*扩展数据块*是对数据库定义的一组连续页。文件扩展名表示该文件中存储的数据的类型。为了在表空间中的所有容器上平均分布数据，表的起始扩展数据块以循环方式分布在所有容器上。如果数据库中包含许多容量较小的表，那么这种扩展数据块分布特别重要。将数据写入多个容器时，建议执行 DB2 条带分割。如果您选择实现磁盘条带分割以及 DB2 条带分割，那么表空间的扩展数据块大小和磁盘的分割大小应该相同。

在 SMS 表空间中，表的空间大小是按需分配的。分配的空间量取决于 *multipage_alloc* 数据库配置参数的设置。如果将此配置参数设置为 YES，那么需要空间时将分配完整的扩展数据块（通常包含两页或更多页）。否则，一次分配的空间将为一页。

缺省情况下，启用了多页文件分配功能。multipage_alloc 数据库配置参数值将指示是否已启用多页文件分配功能。

**注:** 多页文件分配功能不适用于临时表空间。

多页文件分配将只影响一个表的数据和索引部分。这意味着 .LF、.LB 和 .LBA 文件并不会一次扩展一个扩展数据块。

在 SMS 表空间中，当将单个容器中的所有空间都分配给表时，就认为该表空间“已满”，即使其他容器中还有剩余空间。仅当数据库分区中没有任何容器时，才能向该数据库分区中的 SMS 表空间添加容器。

**注:** SMS 表空间可以利用文件系统预取和高速缓存的功能。

SMS 表空间是在 CREATE DATABASE 命令或 CREATE TABLESPACE 语句中使用 MANAGED BY SYSTEM 选项定义的。当设计 SMS 表空间时，必须考虑两个关键要素:

- 表空间的容器。

  必须指定要用于表空间的容器数。标识要使用的所有容器是非常重要的，因为您不能在创建了 SMS 表空间之后添加或删除容器。在分区数据库环境中，在将新数据库分区添加至 SMS 表空间的数据库分区组时，可以使用 ALTER TABLESPACE 语句将容器添加至新的数据库分区。

用于一个 SMS 表空间的每个容器都标识一个绝对或相对目录名。其中每一个目录都可以位于不同的文件系统（物理磁盘）上。表空间的最大大小可以按以下方法估计:

```
容器数 * (操作系统支持的最大文件系统大小)
```

此公式假定有一个唯一的文件系统映射至每个容器，且每个文件系统都具有最大量的可用空间。实际上，情况可能不是这样，表空间最大大小可能小得多。对于数据库对象的大小也有 SQL 限制，它可能影响表空间的最大大小。

**注**: 定义容器时必须很小心。如果容器上已有文件或目录，将返回一个错误（SQL0298N）。

- 表空间的扩展数据块大小。

  扩展数据块大小只能在创建表空间时指定。因为以后不能更改它，因此为扩展数据块大小选择一个适当的值就很重要。

  如果在创建表空间时未指定扩展数据块大小，那么数据库管理器将使用缺省扩展数据块大小来创建表空间，该缺省大小由 *dft_extent_sz* 数据库配置参数定义。此配置参数最初是根据创建该数据库时提供的信息设置的。如果未在 CREATE DATABASE 命令中指定 *dft_extent_sz* 参数，那么会将缺省扩展数据块大小设置为 32。

要为表空间的容器数和扩展数据块大小选择适当的值，必须了解:

- 操作系统对逻辑文件系统的大小施加的限制。

  例如，某些操作系统有 2GB 的限制。因此，如果想要一个 64GB 的表对象，那么在此类型的系统上将需要至少 32 个容器。

  当创建该表空间时，可以指定位于不同文件系统上的容器，以便增加可以存储在该数据库中的数据量。

- 数据库管理器如何管理与一个表空间相关的数据文件和容器。

  在为该表空间指定的第一个容器中创建第一个表数据文件（SQL00002.DAT），并允许此文件增大至该扩展数据块大小。在它达到此大小后，数据库管理器将数据写入下一个容器中的 SQL00002.DAT。此过程将继续，直到所有容器都包含 SQL00002.DAT 文件为止，那时数据库管理器会返回至第一个容器。此过程（称为*条带分割*）会继续在容器目录中运行，直到一个容器装满为止（SQL0289N）或操作系统中不再有空间可分配为止（磁盘已满错误）。条带分割适用于块映射文件（SQLnnnnn.BKM）、索引对象以及用来存储表数据的其他对象。

  **注**: 只要任何一个容器已满，SMS 表空间就满了。因此，每个容器具有相同容量的可用空间是很重要的。

  为了帮助将数据更加均匀地分布到这些容器中，数据库管理器根据将表标识（以上示例中为 SQL00002.DAT）除以容器数所得的余数来确定首先使用的容器。容器从 0 开始依次编号。

## 数据库管理的空间

在 DMS（数据库管理的空间）表空间中，数据库管理器控制存储空间。

存储模型由有限数目的设备或文件组成，这些设备或文件的空间由数据库管理器管理。数据库管理员决定使用哪些设备和文件，并管理这些设备和文件上的空间。这种表空间实质上实现了一种特殊用途的文件系统，用于最好地满足数据库管理器的需要。

DMS 表空间与 SMS 表空间之间的差异在于，对于 DMS 表空间，空间是在创建表空间时分配的。对于 SMS 表空间，空间是根据需要分配的。包含用户定义的表和数据的 DMS 表空间可以定义为存储任何表数据或索引数据的*常规*表空间或*大型*表空间。

在设计 DMS 表空间和容器时，应该考虑下列事项:

- 数据库管理器使用条带分割来确保数据均匀地分布在所有容器上。此操作将数据均匀分布在表空间中的所有容器上，并以循环方式将表的扩展数据块分布在所有容器上。将数据写入多个容器时，建议执行 DB2 条带分割。如果您选择实现磁盘条带分割以及 DB2 条带分割，那么表空间的扩展数据块大小和磁盘的分割大小应该相同。
- 常规表空间的最大大小是 512 GB，每页的大小为 32 KB。大型表空间的最大大小为 16 TB。有关其他页大小的常规表空间的最大大小，请参阅 SQL 和 XML 限制。
- 与 SMS 表空间不同，组成一个 DMS 表空间的容器不必大小相同；但是，通常建议不要这样做，因为会导致在容器间不均匀地进行条带分割，并且会降低性能。如果任何容器已满，DMS 表空间会使用其他容器中的可用空间。
- 因为空间是预分配的，所以在能够创建表空间之前，该空间必须是可用的。当使用设备容器时，该设备也必须有足够的空间以便存储容器定义。每个设备上只能定义一个容器。为避免浪费空间，设备的大小应该等于容器的大小。例如，如果分配给该设备 5000 页，而将设备容器定义为分配 3000 页，那么设备上的 2000 页将是不可用的。
- 在缺省情况下，每个容器中都保留一个扩展数据块作为开销。只使用整个扩展数据块，因此为了对空间进行最优管理，可以使用如下公式来帮助您确定当分配容器时要使用的适当大小:

  ```
  extent_size * (n + 1)
  ```

  其中，*extent_size* 是表空间中每个扩展数据块的大小，而 *n* 是您要在该容器中存储的扩展数据块数目。
- DMS 表空间的最小大小是五个扩展数据块。试图创建小于五个扩展数据块的表空间将产生错误（SQL1422N）。
  - 表空间中有三个扩展数据块是保留给开销使用的。
  - 要存储任何用户表数据，至少需要两个扩展数据块。（这些扩展数据块是一个表的规则数据所必需的，但不是任何索引、长字段或大对象数据所需的，它们需要自己的扩展数据块。）
- 设备容器必须使用带“字符型特殊接口”的逻辑卷，而不是物理卷。
- 在 DMS 表空间中可以使用文件来代替设备。版本 9.5 中的缺省表空间属性 NO FILE SYSTEM CACHING 允许文件对设备执行关闭操作，这样做的好处是不需要设置设备。有关更多信息，请参阅第 160 页的『不使用文件系统高速缓存的表空间』。
- 如果工作负载涉及到 LOB 或 LONG VARCHAR 数据，那么可通过文件系统高速缓存改进性能。

  **注:**
  - 数据库管理器的缓冲池不缓冲 LOB 和 LONG VARCHAR。

  - 继续支持 LONG VARCHAR 和 LONG VARGRAPHIC 数据类型，但不推荐使用这两种类型，将来的发行版中可能会将其除去。
- 某些操作系统允许拥有大小超过 2GB 的物理设备。应该考虑将物理设备划分为多个逻辑设备，以使任何容器都不超过操作系统允许的大小。

**注**: 与 SMS 表空间相似，DMS 文件容器可以利用文件系统预取和高速缓存。但是，使用原始设备容器的 DMS 表空间却不能。

当使用 DMS 表空间时，有两个容器选项: 原始设备和文件。当使用文件容器时，数据库管理器在创建表空间时分配整个容器。这种一开始就分配整个表空间的结果是，即使由文件系统执行分配，物理分配也通常（但不保证）是连续的。当使用原始设备容器时，数据库管理器控制整个设备，并始终确保扩展数据块中的页是连续的。

当使用 DMS 表空间时，您应考虑将每个容器与不同的磁盘相关联。这使表空间容量可以更大，并且能够利用并行 I/O 操作。

CREATE TABLESPACE 语句在数据库中创建新的表空间，向表空间分配容器，并在目录中记录表空间定义和属性。当创建表空间时，扩展数据块大小被定义为许多连续页。扩展数据块是表空间中的空间分配单位。只有一个表或对象（例如，索引）能够使用任何一个扩展数据块中的页。将逻辑表空间地址映射中的扩展数据块分配给表空间中创建的所有对象。扩展数据块分配通过空间映射页进行管理。

逻辑表空间地址映射中的第一个扩展数据块是包含内部控制信息的表空间的头部。第二个扩展数据块是表空间的 SMP 的第一个扩展数据块。SMP 扩展数据块以固定间隔方式分布在表空间中。每个 SMP 扩展数据块都是当前 SMP 扩展数据块到下一 SMP 扩展数据块的扩展数据块位映射。位映射用来跟踪正在使用哪些中间扩展数据块。

SMP 后面的一个扩展数据块是表空间的对象表。对象表是一个内部表，它跟踪表空间中存在哪些用户对象，以及它们的第一个扩展数据块映像页（EMP）扩展数据块的位置。每个对象都有其自己的 EMP，它提供了指向该对象的每一页的映射，这些映射存储在逻辑表空间地址映射中。第 144 页的图 8 说明了如何在逻辑表空间地址映射中分配扩展数据块。

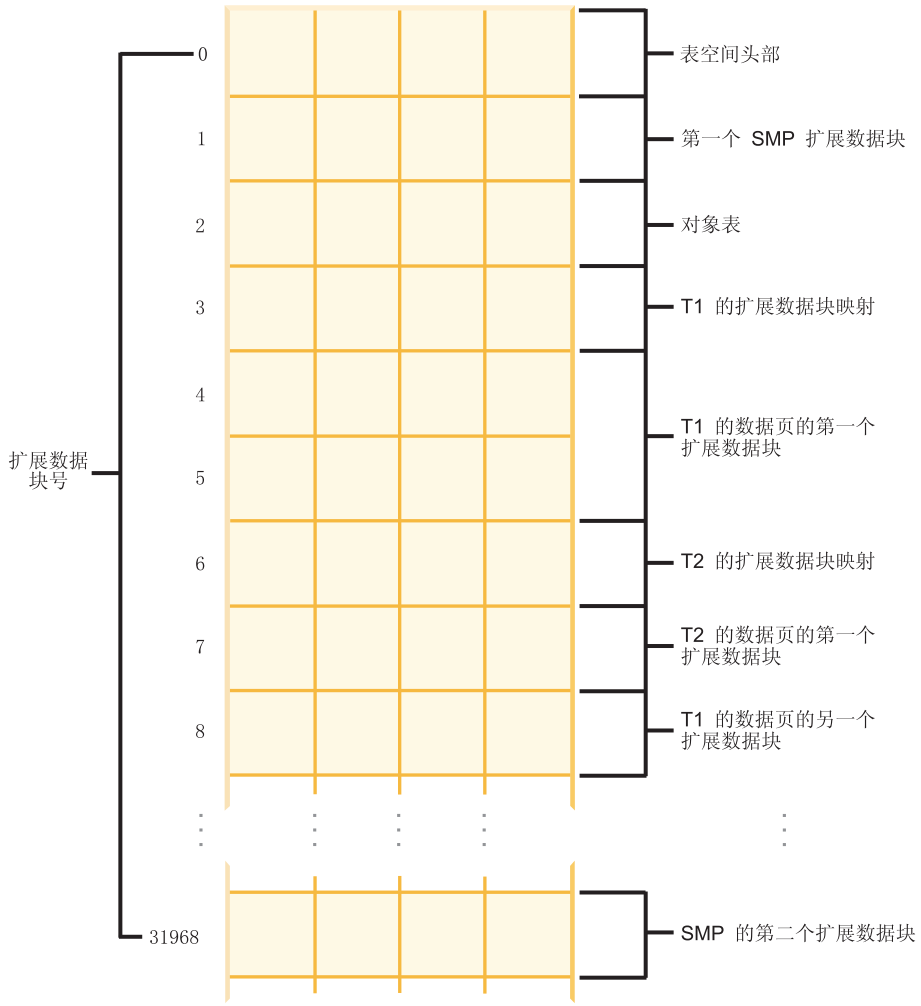



*图 8. 逻辑表空间地址映射*

***DMS 表空间映射:***

表空间映射是数据库管理器的 DMS 表空间的内部表示，它描述表空间中页位置的逻辑至物理转换。本主题描述表空间映射为何有用，以及表空间映射中的信息从何而来。

在分区数据库中，DMS 表空间中的页按照从 0 到 (N-1) 进行逻辑编号，其中 N 是表空间中可用页的数目。

DMS 表空间中的页分组为扩展数据块（基于扩展数据块大小），并且从表空间管理的角度来看，所有对象分配都是以扩展数据块为基础完成的。即，表可能仅使用扩展数据块中的一半页，但是认为整个扩展数据块都在使用中，并且由该对象所有。缺省情况下，使用一个扩展数据块来存放容器标记，并且此扩展数据块中的页不能用来存放数据。但是，如果 DB2_USE_PAGE_CONTAINER_TAG 注册表变量已打开，那么仅将一页用作容器标记。

图 9 显示了 DMS 表空间的逻辑地址映射。

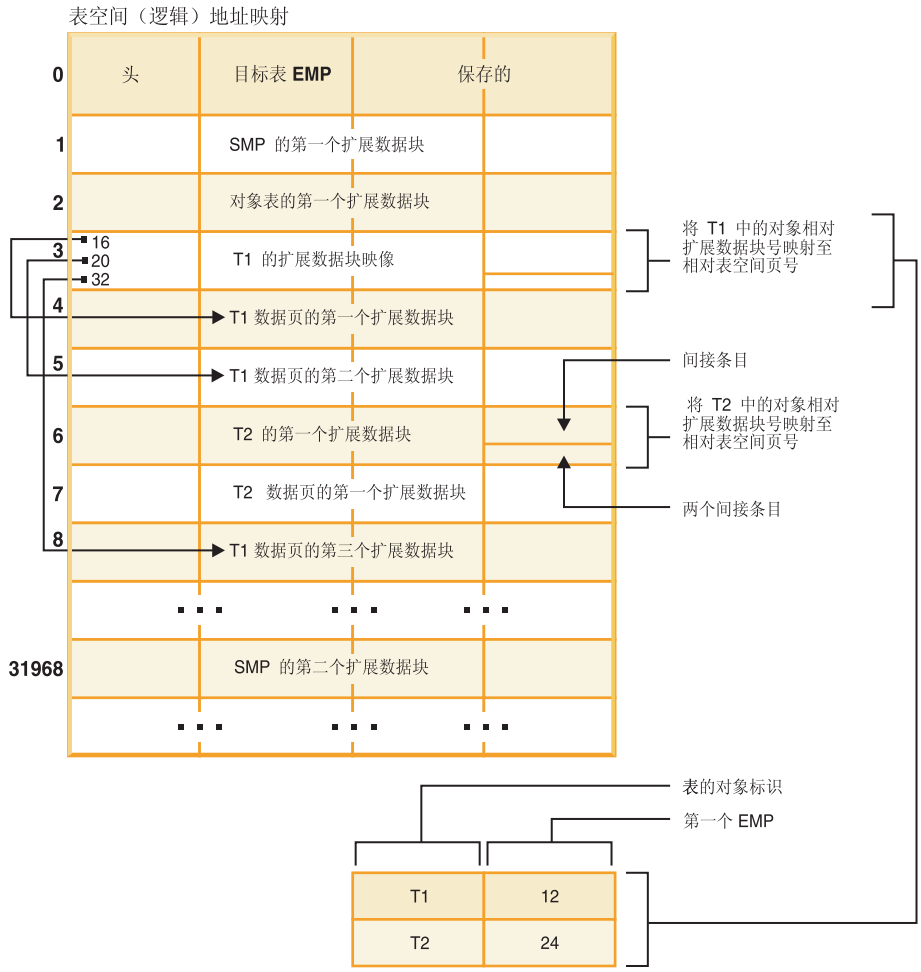



*图 9. DMS 表空间*

在表空间地址映射中，有两种类型的映射页：扩展数据块映像页（EMP）和空间映射页。

对象表是一个内部关系表，它将对象标识映射至该表的第一个 EMP 扩展数据块的位置。这个 EMP 扩展数据块直接或间接地映射出该对象中的所有扩展数据块。每个 EMP 都包含一连串的条目。每一条目都将对象相对扩展数据块号映射至该对象扩展数据块所在的相对表空间页号。直接 EMP 条目直接将对象相对地址映射至相对表空间地址。第一个 EMP 扩展数据块中的最后一个 EMP 页包含间接条目。间接 EMP 条目映射至 EMP 页，EMP 页然后映射至对象页。第一个 EMP 扩展数据块中最后一个 EMP 页中的最后 16 条目包含双重间接条目。

逻辑表空间地址映射中的扩展数据块以循环顺序在与该表空间相关联的容器上进行条带分割。

因为容器中的空间是按扩展数据块来分配的，所以不会使用未组成完整扩展数据块的页。例如，如果您具有 205 页的容器，且扩展数据块大小为 10，那么 1 个扩展数据块将用于存放标记，19 个扩展数据块将用于存放数据，并且剩余的 5 页将被浪费。

如果 DMS 表空间包含单个容器，那么从逻辑页编号到磁盘上的物理位置的转换是直接进行的过程，其中 0、1 和 2 将以磁盘上的相同顺序定位。

当有多个容器并且每个容器大小相同时，它也是相当直接的过程。表空间（包含页 0 到（扩展数据块大小 - 1））中第一个扩展数据块将位于第一个容器中，第二个扩展数据块将位于第二个容器中，依此类推。在最后一个容器之后，过程将从第一个容器开始重复。这种循环过程可以保持数据平衡。

对于包含不同大小容器的表空间，不能使用每个容器轮流进行的简单方法，因为它不会利用大型容器中的额外空间。这就是引入表空间映射的位置 - 它规定如何在表空间内定位扩展数据块，确保物理容器中的所有扩展数据块都可用。

**注:** 在下列示例中，容器大小未将容器标记计算在内。容器大小很小，仅用于说明目的，它们不是建议的容器大小。这些示例显示表空间中不同大小的容器，但建议使用相同大小的容器。

示例 1:

表空间中有 3 个容器，每个容器包含 80 个可用的页，并且表空间的扩展数据块大小是 20。因此，每个容器有 4 个扩展数据块（80 / 20），总数为 12 个扩展数据块。这些扩展数据块位于磁盘上，如图 10 中所示。

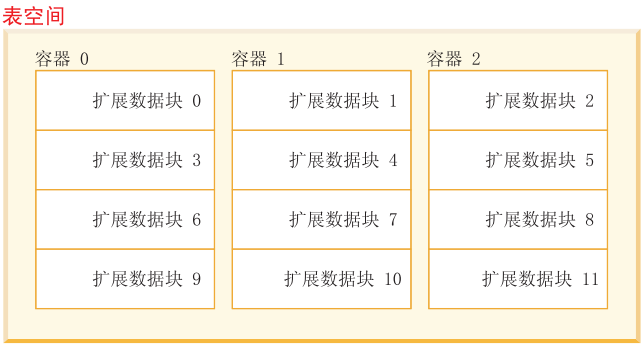



*图 10. 带有三个容器和 12 个扩展数据块的表空间*

要查看表空间映射，使用快照监视器获取表空间快照。在示例 1 中，三个容器大小相等，表空间映射为如下所示:

```
范围    分割    分割区    最大          最大     起始      结束     调节    容器
编号      集     偏移   扩展数据块       页     分割区    分割区
  [0]    [0]       0        11          239       0        3        0    3 (0, 1, 2)
```

*范围*是其中连续范围的分割区包含同一组容器的一段映射。在示例 1 中，所有的分割区（0 到 3）都包含同一组 3 个容器（0、1 和 2），因此认为它是单个范围。

表空间映射中的标题是“范围编号”、“分割集”、“分割区偏移”、“根据范围寻址的最大扩展数据块编号”、“根据范围寻址的最大页编号”、“起始分割区”、“结束分割区”、“范围调节”和“容器列表”。这些将会在示例 2 中更详细地加以说明。

还可以对此表空间进行图解（如图 11 中所示），其中每条竖线对应一个容器，每条横线都称为*分割区*，并且每个单元编号对应一个扩展数据块。

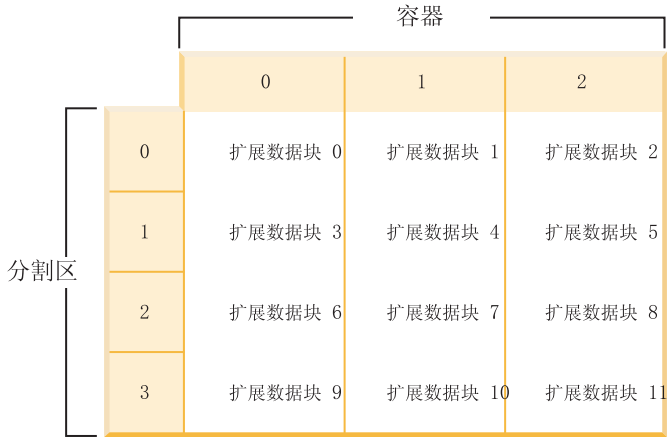



*图 11. 带有三个容器和 12 个扩展数据块，突出显示分割区的表空间*

示例 2:

表空间中有两个容器：第一个大小是 100 页，第二个大小是 50 页并且扩展数据块大小是 25。这意味着第一个容器具有四个扩展数据块并且第二个容器具有两个扩展数据块。可以对表空间进行图解，如图 12 中所示。

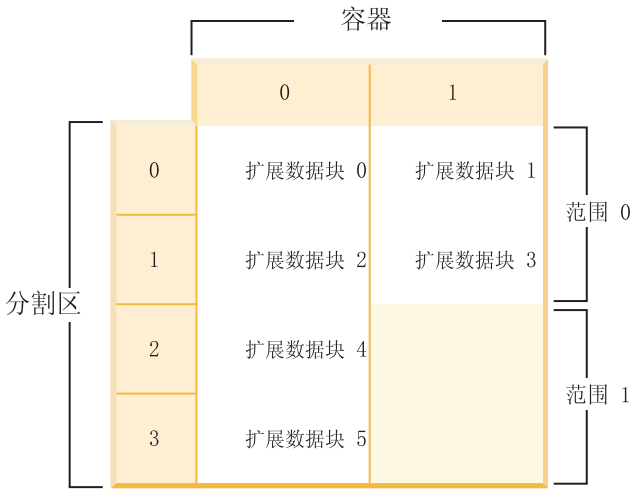



*图 12. 带有两个容器，突出显示范围的表空间*

分割区 0 和 1 包含两个容器（0 和 1），但是分割区 2 和 3 只包含第一个容器（0）。这些分割集的每一个分割集都是一个范围。表空间映射，如表空间快照中所示，类似如下:

```
范围   分割   分割区    最大        最大    起始     结束    调节   容器
编号    集     偏移  扩展数据块      页    分割区   分割区
  [0]    [0]    0          3      99        0        1       0    2
                                                                (0 和 1)
  [1]    [0]    0          5     149     2        3        0     1(0)
```

在第一个范围中有四个扩展数据块，因此在此范围中寻址的最大扩展数据块编号（最大扩展数据块）为 3。每个扩展数据块都有 25 页，因此在第一个范围中有 100 页。因为页的最大编号也是从 0 开始，所以此范围中寻址的最大页编号（最大页）是 99。此范围中的第一个分割区（起始分割区）是 0 并且最后一个分割区（结束分割区）是 1。此范围中有两个容器，它们是 0 和 1。分割区偏移是分割集中的第一个分割，因为只有一个分割集，所以在此情况下它是 0。范围调节（Adj.）是在表空间中重新平衡数据时使用的偏移量。（当在表空间中添加或删除空间时会发生重新平衡。）当没有重新平衡时，它将始终为 0。

在第二个范围中有两个扩展数据块并且因为先前范围中寻址的最大扩展数据块编号是 3，所以此范围中寻址的最大扩展数据块编号是 5。在第二个范围中有 50 页（2 个扩展数据块 * 25 页）并且由于在先前范围中寻址的最大页编号是 99，所以在此范围中寻址的最大页编号是 149。此范围从分割区 2 开始并且在分割区 3 结束。

## 自动存储器表空间

当在未启用自动存储器的数据库中创建表空间时，必须指定 MANAGED BY SYSTEM 或 MANAGED BY DATABASE 子句. 使用这些子句会导致分别创建系统管理的空间（SMS）表空间或数据库管理的空间（DMS）表空间。在两种情况下都必须提供显式的容器列表。

如果数据库启用了自动存储器，那么存在其他选项：您可以指定 MANAGED BY AUTOMATIC STORAGE 子句，或者省略 MANAGED BY 子句（这意味着将使用自动存储器）. 在这种情况下，您不需要提供容器定义，因为数据库管理器会自动指定容器。

以下是一些创建自动存储器表空间的示例语句:

```
CREATE TABLESPACE TS1
CREATE TABLESPACE TS2 MANAGED BY AUTOMATIC STORAGE
CREATE TEMPORARY TABLESPACE TEMPTS
CREATE USER TEMPORARY TABLESPACE USRTMP MANAGED BY AUTOMATIC STORAGE
CREATE LONG TABLESPACE LONGTS
```

虽然自动存储器表空间类型好像是另外的表空间类型，但它实际上只是现有 SMS 和 DMS 类型的扩展。如果您要创建一个常规表空间或大型表空间，那么会将它创建为具有文件容器的 DMS 表空间。如果您要创建一个用户或系统临时表空间，那么会将它创建为具有目录容器的 SMS 表空间。

**注:** 此行为在数据库管理器的将来版本中可能会有所不同。

与这些容器相关联的名称具有以下格式:

```
storage path/instance name/NODE####/database name/T#######/C#######.EXT
```

其中:

*storage path*
是与数据库相关联的存储路径

*instance name*
是创建了数据库的实例

*database name*
是数据库的名称

**NODE####**
是数据库分区号（例如，NODE0000）

**T#######**
是表空间标识（例如，T0000003）

**C#######**
是容器标识（例如，C0000012）

*EXT* 是基于要存储的数据类型的扩展名:

**CAT** 系统目录表空间

**TMP** 系统临时表空间

**UTM** 用户临时表空间

**USR** 用户或常规表空间

**LRG** 大型表空间

## 常规和大型自动存储器表空间与 DMS 表空间之间的区别

常规和大型自动存储器表空间是作为 DMS 表空间创建的，与 DMS 表空间相关联的所有规则和行为都适用。但是，存储器的管理方式存在一些差异，如下表中所示:

*表 44. 管理非自动存储器和自动存储器表空间时存在的差异*

| 非自动存储器 | 自动存储器 |
|---|---|
| 创建表空间时必须显式提供容器列表。 | 创建表空间时不能提供容器列表；但是，数据库管理器将自动指定和分配容器。 |
| 缺省情况下关闭自动重新调整表空间的大小（AUTORESIZE 设置为 NO）。 | 缺省情况下打开自动重新调整表空间的大小（AUTORESIZE 设置为 YES）。 |
| 不能使用 INITIALSIZE 子句来指定表空间的初始大小。 | 可以使用 INITIALSIZE 子句来指定表空间的初始大小。 |
| 可以使用 ALTER TABLESPACE 语句（指定 ADD、DROP 和 BEGIN NEW STRIPE SET 等等）来执行容器操作。 | 不能执行容器操作，因为数据库管理器将管理空间。 |
| 可以使用重定向复原操作来重新定义与表空间相关联的容器。 | 不能使用重定向复原操作来重新定义与表空间相关联的容器，因为数据库管理器将管理空间。 |

正如前面表中所提到的那样，当您创建常规或大型自动存储器表空间时，可以指定初始大小作为 CREATE TABLESPACE 语句的一部分，如以下示例中所示:

```
CREATE TABLESPACE TS1 INITIALSIZE 100 M
```

如果未指定初始大小，那么数据库管理器使用缺省值（即，32MB）。

要创建具有给定大小的表空间，数据库管理器将在存储路径中创建文件容器。如果各路径之间空间分布不均匀，那么创建容器时它们可能具有不同的大小。因此，所有存储路径上的可用空间大致相等非常重要。

如果对表空间启用自动调整大小，那么在使用表空间中的空间时，数据库管理器会自动扩展现有容器并添加新的容器（使用分割集）。无论是扩展还是添加容器，都不会进行重新平衡。

## 临时表空间

系统临时表空间存放数据库管理器在执行诸如排序或连接之类的操作时所需的临时数据。

这些类型的操作需要额外的空间来处理结果集。数据库必须至少有一个系统临时表空间，其页大小与目录表空间相同；在缺省情况下，创建数据库时会创建一个名为 TEMPSPACE1 的系统临时表空间。IBMTEMPGROUP 是此表空间的缺省数据库分区组。

用户临时表空间存放使用 DECLARE GLOBAL TEMPORARY TABLE 语句创建的表的临时数据。为了能够定义已声明临时表，至少一个用户临时表空间应该是使用相应 USE 特权创建的。USE 特权是使用 GRANT 语句授予的。缺省情况下，用户临时表空间不是在创建数据库时创建的。

建议您定义单个临时表空间，使它的页大小等于大多数用户表空间所使用的页大小。这应该适用于典型的环境和工作负载。但最好用不同的临时表空间配置和工作负载进行实验。应该考虑下列几点:

- 在大多数情况下，临时表空间是按顺序以批处理方式访问的。也就是说，插入一组行或者按顺序访存一组行。因此，页大小越大，通常会获得更高的性能，这是因为读取给定的数据量时需要发出的逻辑和物理页请求更少。
- 当使用临时表空间重组表时，该临时表空间的页大小必须与该表的页大小匹配。因此，应确保为现有表使用的每个不同的页大小定义了临时表空间，这样才能使用临时表空间来重组这些表。

  也可以不使用临时表空间来进行重组，即直接在同一表空间中重组该表。当然，这种重组要求在表的表空间中有额外的空间用来完成重组过程。
- 当使用 SMS 系统临时表空间时，您可能想考虑使用注册表变量 DB2_SMS_TRUNC_TMPTABLE_THRESH。当删除时，为系统临时表创建的文件将被截断为大小为 0。可以使用 DB2_SMS_TRUNC_TMPTABLE_THRESH 来避免访问文件系统，从而潜在地让文件大小不为 0，以避免反复扩展和截断文件对性能造成的影响。
- 通常，当存在使用不同页大小的临时表空间时，优化器将选择其缓冲池容纳的行数最多（大多数情况下就是最大的缓冲池）的临时表空间。在这种情况下，比较聪明的做法是给一个临时表空间分配一个足够大的缓冲池，而给其余临时表空间分配较小的缓冲池。这种缓冲池分配将有助于保证有效利用主存储器。例如，如果目录表空间使用 4 KB 页，而其余表空间使用 8 KB 页，那么最佳临时表空间配置可能是:具有一个大缓冲池的一个 8 KB 临时表空间和一个具有小一些的缓冲池的一个 4 KB 表空间。
- 一般情况下，定义具有相同页大小的多个临时表空间没有什么好处。

*确保系统临时表空间的页大小符合要求:*

更大记录标识符（RID）的使用增加了来自查询或定位更新的结果集的行大小。如果结果集中的行大小接近于现有系统临时表空间的最大行长度限制，那么可能需要创建具有更大页大小的系统临时表空间。

**先决条件**

确保具有 SYSCTRL 或 SYSADM 权限来在必要时创建系统临时表空间。

**过程**

要确保系统临时表空间的最大页大小对于查询或定位更新足够大。

1. 确定来自查询或定位更新的结果集的最大行大小。使用曾用来创建表的 DDL 语句来监控查询或者计算最大行大小。
2. 使用 LIST TABLESPACES 命令列出表空间，如以下示例所示:

```
    db2 LIST TABLESPACES SHOW DETAIL...
表空间标识                          = 1
名称                                = TEMPSPACE1
类型                                = 系统管理的空间
内容                                = 系统临时数据
状态                                = 0x0000
    详细说明:
       正常  总页数                              = 10
可用的页数                          = 10
使用的页数                          = 10
空闲页数                            = 不适用
高水位标记（页数）                  = 不适用
页大小（字节数）                    = 4096
扩展数据块大小（页数）              = 32
预取大小（页数）                    = 320
容器数                              = 10
...
```

   可以通过在输出中查找其“内容”字段的值为“系统临时数据”的表空间来识别系统临时表空间。注意每个系统临时表空间的页大小，以及已在其中创建了查询或更新时所引用表的表空间的页大小。

3. 检查结果集中的最大行大小是否适合系统临时表空间的页大小:

```
   maximum_row_size > maximum_row_length - 8 字节（单一分区中
                                                的结构开销）
maximum_row_size > maximum_row_length - 16 字节（DPF 中的结构开销）
```

   其中 maximum_row_size 是结果集的最大行大小，maximum_row_length 是基于所有系统临时表空间的最大页大小所允许的最大长度。查看*SQL Reference, Volume 1*中的“SQL 限制和 XML 限制”来确定每个表空间页大小的最大行长度。

   如果最大行大小小于计算的值，那么查询将以它们在 DB2 UDB 版本 8 中的运行方式运行，且您不必继续此任务。

4. 创建一个系统临时表空间，其大小应至少比创建了表的表空间页大小大一个页大小（如果还没有这样的大小的系统临时表的话）。例如，在 Windows 操作系统上，如果在一个具有 4KB 页大小的表空间中创建了表，那么使用 8KB 的页大小创建其他系统临时表空间:

```
db2 CREATE SYSTEM TEMPORARY TABLESPACE tmp_tbsp
              PAGESIZE 8K
           MANAGED BY SYSTEM
           USING ('d:\tmp_tbsp','e:\tmp_tbsp')
```

如果表空间页大小是 32 KB，那么可以减少在查询中选择的信息或者分开这些查询以适合系统临时表空间页。例如，如果选择了表的所有列，那么可以改为仅选择真正需要的列或者选择某些列的一个子串来避免超出页大小限制。

## SMS 和 DMS 表空间的比较

在确定应使用哪种类型的表空间来存储数据时，需要考虑一些问题。

*SMS 表空间的优点:*

- 直到有需要时，系统才会分配空间
- 由于不必预定义容器，所以创建表空间需要的初始工作较少
- 对范围分区数据创建的索引可以与表数据存储在不同的表空间中

*DMS 表空间的优点:*

- 通过使用 ALTER TABLESPACE 语句，可添加或扩展容器来增加表空间的大小。现有数据可以自动在新的容器集合中重新平衡，以保持最佳 I/O 效率。
- 根据存储的数据的类型，一个表可以分布在多个表空间中:
  - 长字段（LF）和大对象（LOB）数据
  - 索引
  - 常规表数据

  通过分隔表数据，可以提高性能或增加每个表存储的数据量。例如，如果您要使用 4 KB 页大小的大型表空间，那么可以有一个包含 2 TB 正规表数据的表、有一个包含 2TB 索引数据的单独表空间和另一个包含 2 TB 长型数据的单独表空间。如果这三种类型的数据存储在一个表空间中，那么总空间将限制为 2 TB。使用较大的页大小将允许您存储更多数据。请参阅相关链接，以获取数据库管理器页大小限制的完整列表。
- 对范围分区数据创建的索引可以与表数据存储在不同的表空间中。
- 可控制数据在磁盘上的位置（如果操作系统允许）。
- 通常，精心调整的一组 DMS 表空间的性能将优于 SMS 表空间。

**注:** 对于性能敏感的应用程序，特别是涉及大量插入操作的应用程序，建议您使用 DMS 表空间。

并且，在这两种类型的表空间上，数据的放置也会有所不同。例如，进行高效表扫描要求扩展数据块中的页在物理上是连续的。对于 SMS，操作系统的文件系统决定了每个逻辑文件页的物理放置位置。根据文件系统上其他活动的级别以及用来确定放置位置的算法的不同，可能会连续分配这些页。但是，对于 DMS，因为数据库管理器直接与磁盘打交道，所以它可以确保这些页在物理上是连续的。

通常，小型个人数据库用 SMS 表空间管理最容易。另一方面，对于不断增长的大型数据库，您可能只希望使用 SMS 表空间用作临时表空间和目录表空间，而使用具有多个容器的单独的 DMS 表空间用于每个表。另外，您可能想将长字段（LF）数据和索引存储在它们自己的表空间中。

如果选择使用带设备容器的 DMS 表空间，那么需要调整和管理您的环境。

## SMS 和 DMS 工作负载注意事项

您的环境中由数据库管理器管理的主要工作负载类型会影响您选择要使用的表空间类型以及要指定的页大小。

联机事务处理（OLTP）工作负载的特征是：事务需要对数据进行随机访问，通常涉及频繁插入或更新活动和通常返回一小组数据的查询。假定访问是随机的，并且是访问一页或几页，那么不太可能发生预取。

使用设备容器的 DMS 表空间在这种情况下表现得最好。如果不需要最大性能，使用文件容器的 DMS 表空间或 SMS 表空间也适合用于 OLTP 工作负载。请注意，在 FILE SYSTEM CACHING 关闭的情况下，将 DMS 表空间与文件容器配合使用在某种程度上相当于 DMS 原始表空间容器。如果期望很少的顺序 I/O 或不期望它，那么 CREATE TABLESPACE 语句中的 EXTENTSIZE 和 PREFETCHSIZE 参数的设置对于 I/O 的效率就显得不重要。但是，使用 *chngpgs_thresh* 配置参数设置足够数目的页清除程序很重要。

查询工作负载的特征是，事务需要对数据进行顺序访问或部分顺序访问，并常常返回大的数据集。使用多个设备容器且每个容器都在单独的磁盘上的 DMS 表空间最有可能提供有效的并行预取。应该将 CREATE TABLESPACE 语句中的 PREFETCHSIZE 参数的值设置为 EXTENTSIZE 参数的值乘以设备容器数之积。此外，可以将预取大小指定为 -1，此时数据库管理器将自动选择合适的预取大小。这允许数据库管理器以并行方式从所有容器中预取。如果容器的数目更改，或需要使预取更多或更少，那么可以使用 ALTER TABLESPACE 语句相应地更改 PREFETCHSIZE 值。

如果文件系统有自己的预取，那么使用文件来替代查询工作负载较合理。这些文件可以是使用文件容器的 DMS 类型或 SMS 类型。注意，如果使用 SMS，那么需要将目录容器映射至单独的物理磁盘，以实现 I/O 并行性。

混合工作负载的目标是：对于 OLTP 工作负载，使单个 I/O 请求尽可能有效率；而对于查询工作负载，最大程度地提高并行 I/O 的效率。

确定表空间页大小的注意事项如下所示:

- 对于执行随机行读写操作的 OLTP 应用程序，通常最好使用较小的页大小，这样不需要的行就不会浪费缓冲池空间。
- 对于一次访问大量连续行的决策支持系统（DSS）应用程序，页大小大一些会比较好，这样就能减少读取特定数目的行所需的 I/O 请求数。
- 更大的页大小可允许您减少索引中的级别数。
- 越大的页，支持的行越长。
- 在缺省的 4 KB 页上，一个表只能有 500 列，而更大的页大小（8 KB、16 KB 和 32 KB）支持 1012 列。
- 表空间的最大大小与表空间的页大小成正比。

## SMS 和 DMS 设备注意事项

在选择将文件系统文件还是设备用于表空间容器时有几个选项要考虑：数据的缓冲以及是否使用 LOB 或 LOG 数据。

- **数据的缓冲**

从磁盘读取的表数据通常可在数据库的缓冲池中找到。在某些情况下，在应用程序实际使用一个数据页之前，可能从缓冲池释放该页，特别在其他数据页需要缓冲池空间时，更是如此。对于使用系统管理的空间（SMS）或数据库管理的空间（DMS）文件容器的表空间，以上文件系统高速缓存可以消除另外将需要的 I/O。

使用数据库管理的空间（DMS）设备容器的表空间不使用文件系统或其高速缓存。因此，您可以增大数据库缓冲池的大小，减小文件系统高速缓存的大小，以修正这样一个事实，即，使用设备容器的 DMS 表空间并未执行双缓冲区。

如果系统级别的监视工具显示：与等价的 SMS 表空间相比，使用设备容器的 DMS 表空间的 I/O 更高，这种差别可能是由于双缓冲区所导致的。

- **使用 LOB 或 LONG 数据**

  当一个应用程序检索 LOB 或 LONG 数据时，数据库管理器不在它的缓冲区中高速缓存该数据，每次应用程序需要其中一个页时，数据库管理器必须从磁盘对其进行检索。但是，如果 LOB 或 LONG 数据存储在 SMS 或 DMS 文件容器中，文件系统高速缓存可提供缓冲，因此也就改善了性能。

  因为系统目录包含一些 LOB 列，所以应该将它们保存在 SMS 表空间或 DMS 文件表空间中。

## 为表选择表空间时的注意事项

确定如何将表映射至表空间时，应考虑表的分布情况、表数据的数量和类型以及管理问题。

### 表的分布

至少应该确保选择的表空间位于具有您想要的分布的数据库分区组中。

### 表中的数据量

如果计划在一个表空间中存储许多小表，那么考虑使用 SMS 充当该表空间。对于小表，DMS 表现在 I/O 和空间管理效率方面的优点就没有那么重要。SMS 的优点（仅在需要时使用）却对小表更具吸引力。如果一个表较大或者您需要更快地访问表中的数据，应考虑具有较小扩展数据块大小的 DMS 表空间。

您可能希望对每个非常大的表都使用单独的表空间，而将所有的小表组合在单个表空间中。这种分隔还允许您根据表空间的使用选择适当的扩展数据块大小。

### 表数据的类型

例如，有的表可能包含不经常使用的历史记录数据；最终用户可能愿意接受较长的响应时间，来等待对此数据执行的查询。在这种情况下，您可能会为历史记录表使用另一个表空间，并将此表空间分配给访问速率较低的较便宜的物理设备。

此外，您也许能够标识某些表，这些表对于使数据快速可用以及您需要快速响应时间是必不可少的。可能要将这些表置于分配给一个快速物理设备的表空间中，这样将有助于支持这些重要的数据需要。

通过使用 DMS 表空间，还可以将表数据分发在三个不同的表空间中：一个存储索引数据；一个存储大对象（LOB）和长字段（LF）数据；一个存储常规表数据。这允许您选择表空间特征和支持最适合该数据的那些表空间的物理设

备。例如，可能会将索引数据置于可找到的最快的设备上，这样性能可显著提高。如果将一个表分布在各 DMS 表空间中，那么在启用前滚恢复时，应考虑一起备份和复原那些表空间。SMS 表空间不支持以此方式将数据分发在所有表空间中。

**管理问题**

某些管理功能可以在表空间级执行，但不能在数据库或表级别执行。例如，备份表空间（而不是数据库）可以帮助您更好地利用时间和资源。它允许频繁地备份带有大量更改的表空间，同时仅偶尔地备份带有少量更改的表空间。

可以复原数据库或表空间。如果不相关的表不共享表空间，就可以选择复原数据库一个较小的部分以降低成本。

一种好方法是将相关的表编组在一组表空间中。这些表可以通过引用约束相关，也可以通过定义的其他业务约束相关。

如果经常需要删除并重新定义特定表，那么应在它自己的表空间中定义该表，因为删除一个 DMS 表空间比删除一个表更有效率。

## 自动重新调整表空间的大小

在自动存储器表空间启用自动重新调整大小的情况下，数据库管理器会通过添加容器的新分割集来自动处理文件系统变满的情况。

数据库系统中可以存在两种类型的表空间: 系统管理的空间（SMS）和数据库管理的空间（DMS）。与 SMS 表空间相关联的容器是文件系统目录，而这些目录中的文件会随着表空间中对象的增多而增大。。文件会逐渐增大，直到达到其中一个容器的文件系统限制或者达到数据库的表空间大小限制（请参阅 SQL 和 XML 限制）。

DMS 表空间由文件容器或原始设备容器组成，它们的大小是在将容器指定给表空间时设置的。当容器中的所有空间都已被使用时，那么认为表空间将满。但是，与 SMS 表空间不同，您可以使用 ALTER TABLESPACE 语句来添加或扩展容器，从而允许将更多的存储空间提供给表空间。DMS 表空间还有一项称为*自动调整大小*的功能: 当可以自动调整大小的 DMS 表空间中的空间被消耗时，数据库系统可能会对该表空间扩展一个或多个文件容器。SMS 表空间具有类似于自动增长的功能，但术语“自动调整大小”专门用于 DMS。

自动重新调整表空间的大小具有下列含义:

- 启用了自动调整大小的表空间具有版本 8.2.1 或更早发行版不能识别的相关元数据。对这些版本尝试使用启用了自动调整大小的表空间的数据库会产生故障（很有可能会返回 SQL0980C 或 SQL0902C 错误）。如果您尝试连接至数据库或者尝试复原数据库，那么可能会发送错误。如果启用了表空间自动重新调整大小，那么对这些表空间禁用自动调整大小功能就会除去元数据，从而允许对版本 8.2.1 或更早发行版使用该数据库。
- 如果禁用自动调整大小功能，后来又启用此功能，那么与 INCREASESIZE 和 MAXSIZE 相关联的值会丢失。
- 不能对使用原始设备容器的表空间使用此功能，也不能将原始设备容器添加至可以自动调整大小的表空间。尝试执行这些操作会产生错误（SQL0109N）。如果需要添加原始设备容器，那么必须首先禁用此功能。
- 重定向复原操作不能更改容器定义以包括原始设备容器。尝试执行这种操作会产生错误（SQL0109N）。

- 由于最大大小限制了数据库管理器自动增大表空间的方式，所以最大大小也限制了您可增大表空间的方式。也就是说，当执行向表空间添加空间的操作时，生成的大小必须小于或等于最大大小。可以使用 ALTER TABLESPACE 语句的 ADD、EXTEND、RESIZE 或 BEGIN NEW STRIPE SET 子句来添加空间。

## 启用和禁用自动调整大小功能

缺省情况下，不会对 DMS 表空间启用自动调整大小功能。以下语句将创建不启用自动调整大小功能的 DMS 表空间:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
    USING (FILE '/db2files/DMS1' 10 M)
```

要启用自动调整大小功能，对 CREATE TABLESPACE 语句指定 AUTORESIZE YES 子句:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
    USING (FILE '/db2files/DMS1' 10 M) AUTORESIZE YES
```

在创建 DMS 表空间之后，还可以使用带有 AUTORESIZE 子句的 ALTER TABLESPACE 语句来启用或禁用自动调整大小功能:

```
ALTER TABLESPACE DMS1 AUTORESIZE YES
ALTER TABLESPACE DMS1 AUTORESIZE NO
```

有另外两个属性 MAXSIZE 和 INCREASESIZE 与自动调整大小的表空间相关联:

### 最大大小（MAXSIZE）

CREATE TABLESPACE 语句的 MAXSIZE 子句定义表空间的最大大小。例如，以下语句创建可增长至 100 兆字节（如果数据库有多个数据库分区，那么是每个数据库分区的大小）的表空间:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
    USING (FILE '/db2files/DMS1' 10 M)
    AUTORESIZE YES MAXSIZE 100 M
```

MAXSIZE NONE 子句指定表空间没有最大限制。表空间会逐渐增大，直到达到文件系统限制或表空间限制（请参阅 SQL 和 XML 限制）。如果您不指定 MAXSIZE 子句，那么在启用了自动调整大小功能的情况下没有最大值限制。

使用 ALTER TABLESPACE 语句来更改已经启用了自动调整大小功能的表空间的 MAXSIZE 值，如下列示例中所示:

```
ALTER TABLESPACE DMS1 MAXSIZE 1 G
ALTER TABLESPACE DMS1 MAXSIZE NONE
```

如果指定了最大大小，那么数据库管理器强制使用的实际值可能会比指定的值略小，这是因为数据库管理器会尝试使容器的增长保持一致。

### 增大大小（INCREASESIZE）

当表空间中没有可用扩展数据块但是请求了一个或多个扩展数据块时，CREATE TABLESPACE 语句的 INCREASESIZE 子句将定义用来增大表空间的空间量。可以将值指定为显式大小或者指定为百分比，如下列示例中所示:

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
    USING (FILE '/db2files/DMS1' 10 M)
    AUTORESIZE YES INCREASESIZE 5 M
```

```
CREATE TABLESPACE DMS1 MANAGED BY DATABASE
    USING (FILE '/db2files/DMS1' 10 M)
    AUTORESIZE YES INCREASESIZE 50 PERCENT
```

百分比值意味着每当需要增大表空间时都要计算增大大小（由 INCREASESIZE 值指定），并且增大量基于增大时表空间大小所占的百分比。例如，如果表空间大小是 20 MB 而 INCREASESIZE 值是 50%，那么表空间第一次将增大 10 MB（增大到 30 MB），下一次将增大 15 MB。

如果在启用自动调整大小功能时未指定 INCREASESIZE 子句，那么数据库管理器将确定要使用的适当值，该值在表空间的生存期内可能会发生变化。与 AUTORESIZE 和 MAXSIZE 一样，可以使用 ALTER TABLESPACE 语句更改 INCREASESIZE 的值。

如果指定了增大大小，那么数据库管理器将使用的实际值可能会与您提供的值稍微有所不同。对所用值进行这种调整是为了使表空间中各容器的增大保持一致。

## 如何扩展表空间

对于可以自动调整大小的表空间，当所有现有空间都已被使用并请求了更多空间时，数据库管理器会尝试增大该表空间的大小。数据库管理器确定可以扩展表空间中的哪些容器以便不需要进行重新平衡。数据库管理器只扩展位于表空间图（该图描述表空间的存储器布局）的最后范围内的那些容器，并且对它们扩展相同的数量。

例如，考虑如下语句:

```
CREATE TABLESPACE TS1 MANAGED BY DATABASE
    USING (FILE 'C:\TS1CONT' 1000, FILE 'D:\TS1CONT' 1000,
                  FILE 'E:\TS1CONT' 2000, FILE 'F:\TS1CONT' 2000)
    EXTENTSIZE 4
    AUTORESIZE YES
```

请记住，数据库管理器将每个容器的一小部分（一个扩展数据块）用于元数据，以下是根据 CREATE TABLESPACE 语句为表空间创建的表空间图。（表空间图是表空间快照输出的一部分。）

表空间图:

```
范围    分割    分割区     最大         最大     起始      结束     调节    容器
编号     集      偏移   扩展数据块      页     分割区    分割区
[   0] [   0]   0        995        3983      0       248       0     4 (0,1,2,3)
[   1] [   0]   0        1495       5983     249      498       0     2 (2,3)
```

表空间图表明标识为 2 或 3 的容器（E:\TS1CONT 和 F:\TS1CONT）是仅有的在图最后面范围内的容器。因此，当数据库管理器自动扩展此表空间中的容器时，它只扩展这两个容器。

**注**: 如果创建表空间时让所有容器的大小都相同，那么图中只有一个范围。在这种情况下，数据库管理器扩展每一个容器。要防止限制为只扩展一小部分容器，创建表空间时使各容器大小相等。

如先前所讨论的那样，可以指定对表空间最大大小的限制，也可以指定值 NONE 以便不限制表空间的增大。如果指定了 NONE 或者无限制，那么上限由文件系统限制或表空间限制定义；数据库管理器不会尝试让表空间大小增大到超过上限。但是，在达到

上限之前，尝试增大容器可能会因为文件系统已满而失败。在这种情况下，数据库管理器不会再增大表空间大小，但是会向应用程序返回“空间不足”情况。解决此情况有两种方法:

- 增大已满文件系统上可用的空间量。
- 对表空间执行一些容器操作，以使这些容器不再位于表空间图的最后。完成此任务的最简易方法是将新的分割集添加至具有一组新容器的表空间，最佳实践是确保所有容器的大小相同。可以使用带有 BEGIN NEW STRIPE SET 子句的 ALTER TABLESPACE 语句来添加新的分割集。通过添加新的分割集，就会将新的范围添加至表空间图。借助新的范围，数据库管理器自动尝试扩展的容器就会处于此新的分割集中，而旧的容器保持不变。

  **注:** 当暂挂用户启动的容器操作或者正在执行后续生新平衡时，会禁用自动调整大小功能，直到落实了操作或重新平衡完成为止。

例如，对于 DMS 表空间，假定一个表空间具有三个大小相同的容器，并且每个容器都位于它自己的文件系统上。当对表空间执行一些操作时，数据库管理器就会自动扩展这三个容器。最后，其中一个文件系统变满了，对应的容器就不能再增大了。如果该文件系统上不能再提供更多可用空间，那么必须对表空间执行容器操作，使得存在问题的容器不再处于表空间图的最后范围内。在这种情况下，您可以添加新的分割集并指定两个两个容器（仍然具有空间的每个文件系统上一个），也可以指定更多或更少容器（再次确保要添加的每个窗口大小一样并且要使用的每个文件系统上有足够的空间）。当数据库管理器尝试增大表空间的大小时，它现在将尝试扩展此新分割集中的容器而不会尝试扩展旧容器。

### 监视

有关对 DMS 表空间自动调整大小的信息是作为表空间监视器快照输出的一部分显示的。增大大小和最大大小值也包括在输出中，如以下样本中所示:

```
 启用自动调整大小                    = Yes 或 No
当前表空间大小（字节）               = ###
最大表空间大小（字节）               = ### 或 NONE
增加大小（字节）                     = ###
增加大小（百分比）                   = ###
上一次成功调整大小的时间             = YYYY/MM/DD HH:MM:SS.SSSSSS
 上一次调整大小尝试失败              = Yes 或 No
```

## 添加或删除容器时调整预取大小

将为使用 DB2 V8.2 和更高版本创建的任何表空间自动设置磁盘中的所有预取的缺省大小。这意味着数据库管理器将根据多个因素（其中包括扩展数据块大小、表空间中的容器数以及存储设备的属性）来计算适当的预取大小。

可以在何种程度并行进行数据的预取是表空间中的容器数（除了其他事项之外）的一个函数。例如，如果您有两个或更多容器，那么可以并行进行每个容器中的预取，这可以提高数据库的总体性能。如果通过在表空间中添加或删除容器来更改其中的容器数，那么您可以高效地预取的数据量可能会更改。例如，如果您添加一个容器，但是预取的扩展数据块数目保持不变，那么您可能未利用在从其他容器中访存数据的同时从该新容器中访存其他数据的机会。在添加或删除容器时，如果相应地调整预取大小，就可以通过更高效地进行 I/O 操作来保持或提高性能。

使用 CREATE TABLESPACE 或 ALTER TABLESPACE 语句时，通过将表空间的 PREFETCHSIZE 设置为 AUTOMATIC，就不需要手动更新预取大小。除非您已经修改了 **dft_prefetch_sz** 配置参数的缺省值，否则 PREFETCHSIZE 的缺省值为 AUTOMATIC。

如果您希望手动指定预取大小，那么可以采用以下三种方式来指定:

- 创建具有特定预取大小的表空间。如果您手动选择预取大小的值，那么需要记住，每当调整与此表空间相关联的容器数时，就应当调整预取大小。
- 当 **dft_prefetch_sz** 数据库配置参数设置为非缺省值 (AUTOMATIC) 时，请在创建表空间时省略预取大小。如果在创建表空间时未显式提及预取大小，那么数据库管理器将检查此参数。如果发现除 AUTOMATIC 之外的值，那么此值是作为缺省预取大小的值。您需要记住，每当调整与表空间相关联的容器数时，必要时就要调整预取大小。
- 使用 ALTER TABLESPACE 语句手动改变预取大小。

手动调整预取大小时，指定与磁盘条带相对应的大小，以获取最佳 I/O 并行性。要手动计算预取大小，请使用以下公式:

*number_of_containers* × *number_of_disks_per_container* × *extent_size*

例如，假定数据库的扩展数据块大小为 8 页，并且有 4 个容器，每个容器都存在于单个物理磁盘中。将预取大小设置为 4 × 1 × 8 = 32 会得到 32 页的总预取大小。将从这 4 个容器的每个容器中同时读取，总共读取 32 页。

如果每个容器有多个物理磁盘，就像有可能每个容器由一个 RAID 阵列组成一样，那么为了获得最佳 I/O 并行性，应确保正确设置 **DB2_PARALLEL_IO** 注册表变量。（请参阅『使用多个物理磁盘的表空间容器的并行 I/O』。）当您添加或删除容器时，如果已经手动设置了预取大小，那么请记住更新预取大小以反映适当的预取大小。例如，假定每 4 个容器位于 RAID 4+1 阵列中，并且已经设置了 **DB2_PARALLEL_IO** 注册表变量，以允许从每个物理磁盘中并行预取。还假定扩展数据块大小为 8 页。要让每个容器读取一个扩展数据块，将预取大小设置为 4 × 4 × 8 = 128 页。

## 使用多个物理磁盘的表空间容器的并行 I/O

将预取请求提交给预取队列之前，会根据表空间中的容器数将这些预取请求分成许多更小的并行预取请求。**DB2_PARALLEL_IO** 注册表变量用来手动覆盖预取请求的并行性。（有时候将它称为*表空间的并行性*）。当 **DB2_PARALLEL_IO** 设置为 NULL 时（NULL 是缺省值），表空间的并行性等于此表空间中的容器数。如果已启用此注册表变量，那么它定义每个容器的物理磁盘数；表空间的并行性等于容器数乘以 **DB2_PARALLEL_IO** 注册表变量中给定的值。例如，如果表空间中有一个容器由一个 RAID 5 磁盘阵列组成，那么设置此参数，以使数据库管理器将执行的单个预取请求变为 5 个并行预取请求（如果未设置此参数）。如果您有两个容器，每个容器位于 RAID 10 阵列中，那么可以设置此参数，以将每个容器的两个预取请求变为 20 个预取，与每个容器相关联的 10 个磁盘中的每个磁盘 1 个预取。

下面是多个用于说明 **DB2_PARALLEL_IO** 注册表变量如何影响预取的并行性的其他示例。假定已经使用 AUTOMATIC 预取大小定义了表空间。

- **DB2_PARALLEL_IO**=NULL
  - 根据下列各项的组合情况从表空间容器中并行预取:

- 每个表空间中的容器数
- 在 CREATE 或 ALTER TABLESPACE 语句以及 **dft_prefetch_sz** 配置参数中对预取指定的大小。

– 未将预取分成更小的、基于每个磁盘的请求。如果有多个物理磁盘与某个容器相关联，那么将不会从单个容器的磁盘中并行预取。

–

- **DB2_PARALLEL_IO**=*
  – 所有表空间对每个容器都使用缺省主轴数 (6)。预取大小是启用并行 I/O 时的六倍。
  – 所有表空间都启用了并行 I/O。预取请求分解成多个较小请求，每个请求等于预取大小除以扩展数据块大小后的值（或等于容器数目乘以主轴数目）。
- **DB2_PARALLEL_IO**=*:3
  – 所有表空间都让每个容器的主轴数为 3。
  – 所有表空间都启用了并行 I/O。
- **DB2_PARALLEL_IO**=*:3,1:1
  – 所有表空间都让每个容器的主轴数为 3（但是表空间 1 除外，它让每个容器的主轴数为 1）。
  – 所有表空间都启用了并行 I/O。

## 不使用文件系统高速缓存的表空间

建议在表空间级别启用或禁用 UNIX、Linux 和 Windows 上的非缓冲 I/O。

这将允许您在特定表空间上启用或禁用非缓冲 I/O，同时避免数据库的物理布局中的任何依赖性。它还允许数据库管理器确定每个文件最适合使用哪种 I/O，缓冲的还是非缓冲的。

NO FILE SYSTEM CACHING 子句用于启用非缓冲 I/O，从而禁用特定表空间的文件高速缓存。一旦启用了非缓冲 I/O，数据库管理器就会根据平台自动确定将使用直接 I/O（DIO）还是并发 I/O（CIO）。由于使用 CIO 可以提高性能，所以只要支持 CIO，数据库管理器就会使用它；没有用于指定要使用哪一个的用户界面。

为了在使用非缓冲 I/O 时获得最大好处，可能需要增大缓冲池的大小。但是，如果启用了自调整内存管理器并且缓冲池大小设置为 AUTOMATIC，那么数据库管理器将自调整缓冲池大小以获得最佳性能。请注意，版本 9 之前未提供此功能。

要禁用或启用文件系统高速缓存，请分别在 CREATE TABLESPACE 或 ALTER TABLESPACE 语句中指定 NO FILE SYSTEM CACHING 或 FILE SYSTEM CACHING 子句。如果未指定任一子句，那么将使用缺省设置。在使用 ALTER TABLESPACE 的情况下，必须先终止与数据库的现有连接，新的高速缓存策略才会生效。

**注:** 如果将某个属性从缺省值更改为 FILE SYSTEM CACHING 或 NO FILE SYSTEM CACHING，那么没有一种机制可用来将它更改回缺省值。

这种启用和禁用文件系统高速缓存的方法对表空间级的 I/O 方式进行控制（是缓冲的还是非缓冲的）。

可以使用 GET SNAPSHOT FOR TABLESPACES 命令来查询文件系统高速缓存子句的当前设置。例如，以下是 DB2 GET SNAPSHOT FOR TABLEPSACES ON db1 输出中的一个片段:

```
表空间名                           = USERSPACE1
表标识                             = 2
表空间类型                         = 数据库管理的空间
表空间内容类型                     = 所有永久数据。大型表空间。
表空间页大小（以字节计）           = 4096
表空间扩展数据块大小（以页计）     = 32
已启用自动预取大小                 = Yes
当前正在使用的缓冲池标识           = 1
下一次启动的缓冲池标识             = 1
使用自动存储器                     = Yes
已启用自动调整大小                 = Yes
文件系统高速缓存                   = No
表空间状态                         = 0x'00000000'
  详细说明:
      正常
表空间预取大小（以页计）           = 32
总页数                             = 256
```

### 在 UNIX、Linux 和 Windows 上启用/禁用非缓冲 I/O 的其他方法

某些 UNIX 平台支持使用 MOUNT 选项在文件系统级禁用文件系统高速缓存。有关更多信息，请参阅操作系统文档。但是，了解在表空间级和在文件系统级禁用文件系统高速缓存的差别很重要。在表空间级，数据库管理器控制哪些文件将使用文件系统高速缓存打开。在文件系统级，位于特定文件系统上的每个文件都不使用文件系统高速缓存打开。某些平台（如 AIX）在您使用此功能之前有一些要求，比如，序列化读写访问。虽然数据库管理器符合这些要求，但是如果目标文件系统包含非 DB2 文件，在启用此功能之前，请参阅操作系统文档以获取任何要求。

**注**: 在版本 8.1 修订包 4 中引入的但现在已不推荐使用的注册表变量 DB2_DIRECT_IO 对所有 SMS 容器（但 AIX JFS2 上的长字段数据、大对象（LOB）数据和临时表空间除外）禁用文件系统高速缓存。在版本 9.1 或更高版本中设置此注册表变量相当于使用 NO FILE SYSTEM CACHING 子句改变所有表空间、SMS 和 DMS。但是，不推荐使用 DB2_DIRECT_IO，在以后的发行版中会除去此变量。应改为在表空间级启用 NO FILE SYSTEM CACHING。

### 在 Windows 上启用/禁用非缓冲 I/O 的其他方法

在以前的发行版中，可以使用性能注册表变量 DB2NTNOCACHE 来对所有 DB2 文件禁用文件系统高速缓存，以便使更多内存可用于数据库，从而增大缓冲池或排序堆。在版本 9.5 中，不推荐使用 DB2NTNOCACHE，将来的发行版中可能会将它除去。DB2NTNOCACHE 和使用 NO FILE SYSTEM CACHING 子句之间的差别在于是否能够有选择地对表空间禁用高速缓存。从版本 9.5 起，由于 NO FILE SYSTEM CACHING 用作缺省值，所以除非显式指定了 FILE SYSTEM CACHING，否则在实例仅包括新近创建的表空间时，不需要设置此注册表变量来禁用整个实例上的文件系统高速缓存。

### 性能注意事项

非缓冲 I/O 主要用于提高性能。但是，在某些情况下，可能由于较小的缓冲池大小和较小的文件系统高速缓存的组合而导致性能降级，但性能降级不限于这种情况。用于提高性能的建议包括:

- 如果未启用自调整内存管理器，那么启用它并使用 ALTER BUFFERPOOL <name> SIZE AUTOMATIC 将缓冲池大小设置为 AUTOMATIC。这将允许数据库管理器自调整缓冲池大小。
- 如果不打算启用自调整内存管理器，那么以 10% 或 20% 作为增量增大缓冲池大小，直到性能提高为止。
- 如果不打算启用自调整内存管理器，那么改变表空间以使用“FILE SYSTEM CACHING”。这将主要禁用非缓冲 I/O 并还原为缓冲 I/O 以进行容器访问。

在生产系统中实施性能调整之前，应在受控环境中对其进行测试。

选择将文件系统文件和设备用于表空间容器时，应考虑文件系统高速缓存，它按如下所示执行:

- 对于 DMS 文件容器（和所有 SMS 容器），操作系统可能会将页高速缓存在文件系统高速缓存中（除非使用 NO FILESYSTEM CACHING 定义表空间）。
- 对于 DMS 设备容器表空间，操作系统不会将页高速缓存在该文件系统高速缓存中。

## 缺省情况下，新表空间容器使用并发 I/O 或直接 I/O

大多数 AIX、Linux、Solaris 和 Windows 平台上新创建的表空间容器的缺省 I/O 机制为 CIO/DIO（并发 I/O 或直接 I/O）。与缓冲 I/O 相比，在具有大量事务处理工作负载和回滚时此缺省 I/O 机制可增大吞吐量。

FILE SYSTEM CACHING 或 NO FILE SYSTEM CACHING 属性指定是否将在文件系统级别高速缓存 I/O 操作:

- FILE SYSTEM CACHING 指定将在文件系统级别高速缓存目标表空间中的所有 I/O 操作。
- NO FILE SYSTEM CACHING 指定所有 I/O 操作将绕过文件系统级别高速缓存。

对于 SMS 表空间，长字段 (LF) 数据和大对象 (LOB) 数据不需要非缓冲 I/O 访问权，即使在设置了 NO FILE SYSTEM CACHING 表空间属性时。缓冲发生在文件系统高速缓存中（取决于操作系统配置和行为），并有可能提高性能。

对于 DMS 表空间，在执行 I/O 时，DB2 不会区分不同类型的数据。除非在启用 FILE SYSTEM CACHING 的情况下配置表空间，否则不发生 LF 或 LOB 数据缓冲。如果出于性能原因，要在 DMS 表空间中缓冲 LF 或 LOB 数据，那么您可以将该数据置于单独的 DMS 表空间中并显式启用 FILE SYSTEM CACHING

下列接口包含 FILE SYSTEM CACHING 属性:

- CREATE TABLESPACE 语句
- CREATE DATABASE 命令
- sqlecrea() API（使用 SQLETSDESC 结构的 *sqlfscaching* 字段）

未在 CREATE TABLESPACE 语句或 CREATE DATABASE 命令中指定此属性时，数据库管理器将使用基于平台和文件系统类型的缺省行为处理请求。请参阅第 163 页的『文件系统高速缓存配置』以了解准确的行为。对于 sqlecrea() API，如果 *sqlfscaching* 字段的值为 0x2，那么它指示数据库管理器使用缺省设置。

请注意，下列工具当前解释 FILE SYSTEM CACHING 属性的值:

- GET SNAPSHOT FOR TABLESPACES 命令

- db2pd –tablespaces 命令
- db2look –d <dbname> –l 命令

对于 db2look，如果未指定 FILE SYSTEM CACHING 属性，那么输出将不包含此属性。

### 示例

假定数据库和所有相关表空间容器位于 AIX JFS 文件系统上并且发出了以下语句:

```
DB2 CREATE TABLESPACE JFS2
```

在先前版本中，如果未指定该属性，那么数据库管理器将使用缓冲 I/O（FILE SYSTEM CACHING）作为 I/O 机制；对于版本 9.5，数据库管理器使用 NO FILE SYSTEM CACHING。

## 文件系统高速缓存配置

缺省情况下，操作系统将高速缓存从磁盘读写的文件数据。

一个典型的读操作涉及以下物理磁盘访问：将数据从磁盘读取到文件系统高速缓存中，然后将这些数据从高速缓存复制到应用程序缓冲区。同样，写操作涉及以下物理磁盘访问：将数据从应用程序缓冲区复制到文件系统高速缓存，然后将它从文件系统缓冲区复制到物理磁盘。在 CREATE TABLESPACE 语句的 FILE SYSTEM CACHING 子句中反映了在文件系统级高速缓存数据的这种行为。由于数据库管理器使用缓冲池管理其自身的数据高速缓存，所以如果适当调整缓冲池大小，就不需要在文件系统级进行高速缓存。

**注**：数据库管理器已通过使高速缓存中的页无效来防止高速缓存大多数 DB2 数据，但 AIX 上的临时数据和 LOB 除外。

由于两次高速缓存需要额外的 CPU 周期，所以在某些情况下，在文件系统级和缓冲池中进行高速缓存可能会导致性能下降。为了避免两次高速缓存，大多数文件系统都有在文件系统级禁用高速缓存的功能。此功能通常称为*非缓冲 I/O*。在 UNIX 上，此功能通常称为*直接 I/O（或 DIO）*。在 Windows 上，此功能相当于使用 FILE_FLAG_NO_BUFFERING 标记打开文件。此外，某些文件系统（例如，IBM JFS2 或 Symantec VERITAS VxFS）也支持增强型直接 I/O，即，高速执行的*并行 I/O（CIO）*功能。数据库管理器通过 NO FILE SYSTEM CACHING 表空间子句支持此功能。设置此 CIO 功能后，数据库管理器自动利用具有 CIO 功能的文件系统上的此项功能。此功能有助于降低文件系统高速缓存的内存需求，从而使得有更多内存用于其他用途。

在版本 9.5 之前，如果未指定 NO FILE SYSTEM CACHING 或 FILE SYSTEM CACHING，那么暗含指定了关键字 FILE SYSTEM CACHING。对于版本 9.5，如果未指定任一关键字，那么使用缺省值 NO FILE SYSTEM CACHING。此更改仅影响新创建的表空间。在版本 9.5 之前创建的现有表空间不受影响。此更改适用于 AIX、Linux、Solaris 和 Windows，但存在下列例外情况，这些情况下的缺省行为保持为 FILE SYSTEM CACHING:

- AIX JFS
- Solaris 非 VxFS 文件系统
- Linux for System z®
- 所有 SMS 临时表空间文件
- SMS 永久表空间文件中的长字段（LF）和大对象（LOB）数据文件

要覆盖缺省设置，请指定 FILE SYSTEM CACHING 或 NO FILE SYSTEM CACHING。

## 受支持的配置

表 45 显示了用于不使用文件系统高速缓存的表空间的受支持配置。它还指示：(a) 每种情况下是使用 DIO 还是增强型 DIO，以及 (b) 未对表空间指定 NO FILE SYSTEM CACHING 和 FILE SYSTEM CACHING 时基于平台和文件系统类型的缺省行为。

*表 45. 不使用文件系统高速缓存的表空间的受支持配置*

| 平台 | 文件系统类型和必需的最低级别 | 指定了 **NO FILE SYSTEM CACHING** 时，由数据库管理器提交的 **DIO** 或 **CIO** 请求 | 未指定 **NO FILE SYSTEM CACHING** 和 **FILE SYSTEM CACHING** 时的缺省行为 |
|---|---|---|---|
| AIX 5.3 和更高版本 | 日志文件系统（JFS） | DIO | FILE SYSTEM CACHING（请参阅注释 1.） |
| AIX 5.3 和更高版本 | 并发日志文件系统（JFS2） | CIO | NO FILE SYSTEM CACHING |
| AIX 5.3 和更高版本 | VERITAS Storage Foundation for DB2 4.1（VxFS）<br><br>VERITAS Storage Foundation for DB2 5.0 (VxFS) | CIO | NO FILE SYSTEM CACHING |
| HP-UX V11iv2 和 11iv3 (Itanium®) | VERITAS Storage Foundation 4.1 (VxFS)<br><br>VERITAS Storage Foundation 5.0 (VxFS)（请参阅注释 6。） | CIO | FILE SYSTEM CACHING |
| Solaris 9 | UNIX 文件系统（UFS） | DIO | FILE SYSTEM CACHING（请参阅注释 2.） |
| Solaris 10 | UNIX 文件系统（UFS） | CIO | FILE SYSTEM CACHING（请参阅注释 2.） |
| Solaris 9 和 Solaris 10 | VERITAS Storage Foundation for DB2 4.1（VxFS）<br><br>VERITAS Storage Foundation for DB2 5.0 (VxFS) | CIO | NO FILE SYSTEM CACHING |
| Linux 分发产品 SLES 9 和更高版本，以及 RHEL 4 和更高版本<br><br>（在这些体系结构上：x86、x64 和 POWER®） | ext2、ext3 和 reiserfs | DIO | NO FILE SYSTEM CACHING |
| Linux 分发产品 SLES 9 和更高版本，以及 RHEL 4 和更高版本<br><br>（在这些体系结构上：x86、x64 和 POWER） | VERITAS Storage Foundation for DB2 4.1（VxFS）<br><br>VERITAS Storage Foundation for DB2 5.0 (VxFS) | CIO | NO FILE SYSTEM CACHING |

*表 45. 不使用文件系统高速缓存的表空间的受支持配置 (续)*

| 平台 | 文件系统类型和必需的最低级别 | 指定了 NO FILE SYSTEM CACHING 时，由数据库管理器提交的 DIO 或 CIO 请求 | 未指定 NO FILE SYSTEM CACHING 和 FILE SYSTEM CACHING 时的缺省行为 |
|---|---|---|---|
| Linux 分发产品 SLES 9 和更高版本，以及 RHEL 4 和更高版本<br><br>（在此体系结构上：zSeries®） | 使用光纤通道协议（FCP）的小型计算机系统接口（SCSI）磁盘上的 ext2、ext3 或 reiserfs | DIO | FILE SYSTEM CACHING |
| Windows | 没有特定要求，在 DB2 支持的所有文件系统上工作 | DIO | NO FILE SYSTEM CACHING |

**注:**

1. 在 AIX JFS 上，FILE SYSTEM CACHING 是缺省值。
2. 在 Solaris UFS 上，NO FILE SYSTEM CACHING 是缺省值。
3. 数据库管理器的 VERITAS Storage Foundation 可能有不同的操作系统先决条件。上面列示的平台是当前发行版支持的平台。有关 DB2 对先决条件信息的支持，请咨询 VERITAS Storage Foundation。
4. 如果使用 SFDB2 5.0 而不是上面的最低级别，那么必须使用 SFDB2 5.0 MP1 RP1 发行版或更高版本。此发行版包括特定于版本 5.0 的修正。
5. 现在，VERITAS Storage Foundation 5.1 在基本产品中包括 CIO 支持，不再需要该产品的 DB 版本。
6. 在 HP 上，使用 5.0.1 OnlineJFS 启用了 CIO，不需要其他 VERITAS 许可证。
7. 如果您不希望数据库管理器对 NO FILE SYSTEM CACHING 选择缺省设置，请在相关的 SQL、命令或 API 中指定 FILE SYSTEM CACHING。

### 示例

**示例 1**: 缺省情况下，将使用非缓冲 I/O 创建新表空间；暗含指定了 NO FILE SYSTEM CACHING 子句。

```
CREATE TABLESPACE table space name...
```

**示例 2**: 在以下语句中，NO FILE SYSTEM CACHING 子句指示对于此特定表空间，文件系统级高速缓存将 OFF。

```
CREATE TABLESPACE table space name ... NO FILE SYSTEM CACHING
```

**示例 3**: 以下语句对现有表空间禁用文件系统级高速缓存:

```
ALTER TABLESPACE table space name ... NO FILE SYSTEM CACHING
```

**示例 4**: 以下语句对现有表空间启用文件系统级高速缓存:

```
ALTER TABLESPACE table space name ... FILE SYSTEM CACHING
```

## 表空间扩展数据块大小

表空间的扩展数据块大小表示在将数据写入下一个容器之前，将写入当前容器的表数据的页数。

当选择扩展数据块大小时，应考虑:

- 表空间中表的大小和类型。

  将 DMS 表空间中的空间一次分配给表一个扩展数据块。当填充该表而一个扩展数据块变满时，会分配新的扩展数据块。保留了 DMS 表空间容器存储器，这意味着将分配新的扩展数据块，直到彻底用完容器为止。

  将 SMS 表空间中的空间一次分配给表一个扩展数据块或者一次分配给表一页。当填充该表而一个扩展数据块或页变满时，会分配新的扩展数据块或页，直到使用了文件系统中的所有扩展数据块或页为止。当使用 SMS 表空间时，允许进行多页文件分配。多页文件分配允许分配扩展数据块而不是一次分配一页。

  缺省情况下，启用了多页文件分配功能。*multipage_alloc* 数据库配置参数值将指示是否已启用多页文件分配功能。

  **注:** 多页文件分配功能不适用于临时表空间。

  一个表由下列单独的表对象组成:

  – 数据对象。它是存储规则列数据的地方。
  – 索引对象。在表上定义的所有索引都存储在这里。
  – 长字段（LF）数据对象。如果表有一个或多个 LONG 列，那么长字段数据存储在此处。
  – 两个大对象（LOB）数据对象。如果表有一个或多个 LOB 列，那么它们都存储在这两个表对象中:
    - 一个表对象用于存储 LOB 数据
    - 第二个表对象用于存储描述 LOB 数据的元数据
  – 多维集群（MDC）表的块映射对象。
  – 一个额外的 XDA 对象，它存储 XML 文档。

  每个表对象都是单独存储的，每个对象按需要分配新的扩展数据块。每个 DMS 表对象还与称为扩展数据块映像的元数据对象配成一对，该元数据对象描述该表空间中属于该表对象的所有扩展数据块。用于扩展数据块映像的空间也是以一次一个扩展数据块的方式分配。因此，DMS 表空间中对象的初始空间分配是两个扩展数据块。（SMS 表空间中对象的初始空间分配是一页。）

  如果您在一个 DMS 表空间中有多个较小的表，那么可能要分配相对大的空间来存储相对少量的数据。在这种情况下，应该指定小的扩展数据块大小。另一方面，如果您有一个增长速率高的非常大的表，且您使用具有较小扩展数据块大小的 DMS 表空间，那么可能会产生与其他扩展数据块的频繁分配相关的不需要的开销。

- 对这些表访问的类型。

  如果对表的访问包括许多查询或处理大量数据的事务，那么从表中预取数据可以显著改善性能。

- 必需的扩展数据块的最小数目。

  如果容器中没有足够的空间以供表空间的五个扩展数据块使用，那么将无法创建表空间。

## 表空间的页大小

设计表空间时，需要考虑页大小。

可以使用 4K、8K、16K 或 32K 页大小限制。其中每个页大小还有最大值，每种表空间类型都必须遵循此最大值。

表 46 显示了不同类型的表空间特定于页大小的限制:

*表 46. 表空间特定于页大小的限制*

| 表空间类型（以千兆字节计） | 4K 页大小限制 | 8K 页大小限制 | 16K 页大小限制 | 32K 页大小限制 |
|---|---|---|---|---|
| SMS 表空间 | 64 | 128 | 256 | 512 |
| DMS 表空间（常规） | 64 | 128 | 256 | 512 |
| DMS 表空间（大型） | 2048 | 4096 | 8192 | 16 384 |
| 自动存储器表空间（常规） | 64 | 128 | 256 | 512 |
| 自动存储器表空间（大型） | 2048 | 4096 | 8192 | 16 384 |
| 临时表空间 | 64 | 128 | 256 | 512 |

有关不同类型的表空间的数据库和索引页大小限制，请参阅 SQL 和 XML 限制中的数据库管理器特定于页大小的限制。

要确保系统临时表空间的最大页大小对于查询或定位更新来说足够大，请参阅《*迁移指南*》中的『确保系统临时表空间的页大小符合要求』。

## 表空间磁盘 I/O

表空间的类型和设计决定了对该表空间执行的 I/O 的效率。

在考虑其他关于表空间设计和使用的问题之前，您应该了解下列概念:

**大块读取**
在单个请求中检索多页（通常为一个扩展数据块）的一种读取。一次读取几页比分别读取每页更有效。

**预取** 在一个查询引用页之前对那些页的读取。总的目的是为缩短响应时间。如果页的预取可以与查询的执行异步发生，就能够达到此目的。当 CPU 或 I/O 子系统以最大能力运行时达到最佳响应时间。

**页清除** 当读取和修改页时，它们会累积在数据库缓冲池中。当读入一页时，便将其读入到缓冲池页中。如果该缓冲池已充满修改的页，那么必须将修改的这些页的其中一页写出至磁盘，然后才能再读入新的页。为避免缓冲池变满，页清除程序代理程序的任务就是写出修改的页，以保证缓冲池页可用于将来的读取请求。

无论何时，只要有利，数据库管理器就会执行大块读取。当检索顺序排列或本质上是部分顺序排列的数据时，通常会发生这种情况。在一个读取操作中读取的数据量取决于扩展数据块大小 - 扩展数据块大小越大，一次可以读取的页就越多。

如果可以将页从磁盘读入缓冲池内的连续页中，那么可以进一步提高顺序预取的性能。因为缺省情况下缓冲池是基于页的，所以从磁盘的连续页中读取时，不能保证会找到一组连续页。基于块的缓冲池可以用于此目的，因为它们不仅包含页区域，还包含可用于几组连续页的块区域。每一组连续页都命名为一个块，并且每个块都包含称为块大小的若干页。页和块区域的大小以及每个块中的页的数目都是可配置的。

扩展数据块存储在磁盘上的方式影响 I/O 效率。在使用设备容器的 DMS 表空间中，数据往往在磁盘上是连续的，且可以在最短的搜索时间和磁盘等待时间内进行读取。如果使用的是文件，那么为了供 DMS 表空间使用而预分配的大文件在磁盘上也往往是连续的，尤其当该文件分配在一个干净的文件空间中时更是这样。但是，数据可能被系统文件分散，并存储在磁盘上的多个位置中。当使用一次将文件扩展一页的 SMS 表空间时（这使产生碎片的概率更高），最可能发生此情况。

可以通过更改 CREATE TABLESPACE 或 ALTER TABLESPACE 语句中的 PREFETCHSIZE 选项来控制预取的程度，或者可以将预取大小设置为 AUTOMATIC 以让数据库管理器自动选择最适合的大小来使用。（该数据库中所有表空间的缺省值由 *dft_prefetch_sz* 数据库配置参数设置。）PREFETCHSIZE 参数告诉数据库管理器在触发预取时要读取的页数。通过在 CREATE TABLESPACE 语句上将 PREFETCHSIZE 设置为 EXTENTSIZE 参数的倍数，可以并行读取多个扩展数据块。（该数据库中所有表空间的缺省值由 *dft_extent_sz* 数据库配置参数设置。）EXTENTSIZE 参数指定在跳至下一个容器之前将写入一个容器的 4 KB 大小的页数。

例如，假定有一个表空间使用三个设备。如果将 PREFETCHSIZE 设置为 EXTENTSIZE 的三倍，那么数据库管理器可以用并行方式从每个设备中执行大块读取，从而显著增大 I/O 吞吐量。此情况假定每个设备是一个单独的物理设备，且控制器具有足够的带宽来处理来自每个设备的数据流。请注意，数据库管理器可能需要根据查询速度、缓冲池利用率和其他因素在运行时动态调整预取参数。

某些文件系统使用它们自己的预取方法（例如，在 AIX 上的“日志文件系统”）。在某些情况下，文件系统的预取设置得比数据库管理器的预取更主动。这可能导致使用文件容器的 SMS 和 DMS 表空间的预取似乎比使用设备的 DMS 表空间的预取执行效率更高。但这是误导，因为它可能是在文件系统中发生的其他级别的预取的结果。DMS 表空间应该能够比任何等价配置的执行效率高。

为提高预取效率（甚至读取效率），必须存在足够数量的干净缓冲池页。例如，可能有一个并行预取请求要从一个表空间读取三个扩展数据块，对于正在读取的每一页，从缓冲池写出经过修改的一页。该预取请求可能被拖慢，导致它跟不上查询的进展。应配置足够数量的页清除程序，以满足预取请求。

## 定义初始表空间

创建数据库时，将定义三个表空间：(1) 用于系统目录表的 SYSCATSPACE，(2) 用于在数据库处理期间创建的系统临时表的 TEMPSPACE1 以及 (3) 用于用户定义的表和索引的 USERSPACE1。

**注：** 当第一次创建一个数据库时，不创建用户临时表空间。

除非另外指定，否则三个缺省表空间由自动存储器管理。

通过使用 CREATE DATABASE 命令，可以指定缺省缓冲池和初始表空间的页大小。此缺省值还表示所有将来 CREATE BUFFERPOOL 和 CREATE TABLESPACE 语句的缺省页大小。如果在创建数据库时不指定页大小，那么缺省页大小是 4 KB。

要使用命令行来定义初始表空间，请输入:

```
CREATE DATABASE <name>
    CATALOG TABLESPACE
        MANAGED BY SYSTEM USING ('<path>')
        EXTENTSIZE <value> PREFETCHSIZE <value>
    USER TABLESPACE
        MANAGED BY DATABASE USING (FILE'<path>' 5000,
                            FILE'<path>' 5000)
        EXTENTSIZE <value> PREFETCHSIZE <value>
    TEMPORARY TABLESPACE
        MANAGED BY SYSTEM USING ('<path>')
    WITH "<comment>"
```

如果不想使用这些表空间的缺省定义，那么可以在 CREATE DATABASE 命令中指定它们的特征。例如，可使用以下命令在 Windows 上创建数据库:

```
CREATE DATABASE PERSONL
    CATALOG TABLESPACE
        MANAGED BY SYSTEM USING ('d:\pcatalog','e:\pcatalog')
        EXTENTSIZE 16 PREFETCHSIZE 32
    USER TABLESPACE
        MANAGED BY DATABASE USING (FILE'd:\db2data\personl' 5000,
                            FILE'd:\db2data\personl' 5000)
        EXTENTSIZE 32 PREFETCHSIZE 64
    TEMPORARY TABLESPACE
        MANAGED BY SYSTEM USING ('f:\db2temp\personl')
    WITH "Personnel DB for BSchiefer Co"
```

在本示例中，显式提供了每个初始表空间的定义。只需要为不希望使用缺省定义的那些表空间指定表空间定义。

**注**: 当您在分区数据库环境中工作时，不能创建容器或将容器指定给特定数据库分区。首先，必须使用缺省用户和临时表空间创建数据库。然后应使用 CREATE TABLESPACE 语句来创建必需的表空间。最后，可删除缺省表空间。

CREATE DATABASE 命令上的 MANAGED BY 短语的编码与 CREATE TABLESPACE 语句上的 MANAGED BY 短语遵循同一格式。

您可以在需要时添加其他用户和临时表空间。不能删除目录表空间 SYSCATSPACE 或创建另一个目录表空间；且必须始终存在至少一个页大小为 4 KB 的系统临时表空间。可以创建其他系统临时表空间。在创建表空间之后，您也不能更改它的页大小或扩展数据块大小。

## 连接 DMS 直接磁盘访问设备

使用容器存储数据时，数据库管理器支持直接磁盘访问（原始 I/O）。

此类型的支持允许您将直接磁盘访问（原始）设备连接至任何 DB2 数据库系统。

创建表空间时，必须知道准备引用的容器的设备名或文件名。必须知道与要分配给表空间的每个设备名或文件名相关联的空间量。需要正确的许可权才能读写容器。

用于标识直接磁盘访问的物理方法和逻辑方法随操作系统不同而不同:

- 在 Windows 操作系统上:

  要指定物理硬盘驱动器，使用以下语法:

  \\.\PhysicalDriveN

  其中 N 表示系统中的一个物理驱动器。在这种情况下，N 可以替换为 0、1、2 或任何其他正整数:

  \\.\PhysicalDrive5

  要指定逻辑驱动器（即，未格式化的数据库分区），使用以下语法:

  \\.\N:

  其中 N: 表示系统中的一个逻辑驱动器盘符。例如，N: 可被 E: 或任何其他驱动器盘符替换。要克服使用盘符来标识驱动器所带来的局限性，可对逻辑驱动器使用全局唯一标识（GUID）。

  对于 Windows，有一种新方法可用来指定 DMS 原始表空间容器。创建卷（即基本磁盘数据库分区或动态卷）时，对其指定了全局唯一标识（GUID）。在表空间定义中指定容器时，可将 GUID 用作设备标识。GUID 在系统间是唯一的，这意味着在多分区数据库中，即使磁盘分区定义相同，每个数据库分区的 GUID 也各不相同。

  工具 *db2listvolumes.exe* 可用来（仅在 Windows 操作系统上）使 Windows 系统上定义的所有磁盘卷的 GUID 显示起来更加容易。此工具在其运行的当前目录中创建两个文件。一个文件称为 `volumes.xml`，包含有关用 XML 编码的每个磁盘卷的信息，以易于在启用了 XML 的浏览器上进行查看。第二个文件称为 `tablespace.ddl`，包含指定表空间容器的必需语法。必须更新此文件才能填写表空间定义所需的余下信息。db2listvolumes 命令不需要任何命令行自变量。
- 在 Linux 和 UNIX 平台上，逻辑卷对用户和应用程序可以单个相邻且可扩展的磁盘卷出现。尽管它以此方式显示，但是，它也可以位于不相邻的物理数据库分区上，甚至可以位于多个物理卷上。逻辑卷还必须包含在单个卷组中。每个卷组最多只能有 256 个逻辑卷。每个卷组最多只能有 32 个物理卷。可以使用 mklv 命令来创建其他的逻辑卷。此命令允许您指定逻辑卷的名称和定义它的特征，包括要为它分配的逻辑分区的数目和位置。

  在创建逻辑卷之后，可以使用 chlv 命令来更改它的名称和特征，并且可以使用 extendlv 命令来增加分配给它的逻辑分区数。在创建时，逻辑卷的缺省最大大小为 512 个逻辑分区，除非您将它指定为更大。chlv 命令可用来重设此限制。

  在 AIX 中，操作系统命令、库子例程以及其他允许您建立和控制逻辑卷存储器的工具的集合称为“逻辑卷管理器”（LVM）。LVM 通过在存储空间的简单而灵活的逻辑视图与实际的物理磁盘之间映射数据来控制磁盘资源。

  有关 mklv 和其他逻辑卷命令以及 LVM 的更多信息，请参阅 *AIX 5L Version 5.2 System Management Concepts: Operating System and Devices*。

## 配置和设置 DMS 直接磁盘访问（Linux）

使用容器来存储数据时，数据库管理器支持使用块设备接口（即原始 I/O）直接访问磁盘（原始）。

在 Linux 上设置原始 I/O 之前，需要一个或多个可用 IDE 或 SCSI 磁盘数据库分区。为了在创建表空间时引用磁盘分区，您必须知道磁盘分区的名称以及与要分配给该表空间的磁盘分区相关联的空间量。

在 Linux 环境中工作时应使用下列信息。在 Linux/390 上，数据库管理器不支持直接磁盘访问设备。

要在 Linux 上配置原始 I/O:

在本示例中，要使用的原始数据库分区是 /dev/sda5。它应该不包含任何有用的数据。

1. 计算此数据库分区中的页数（每页 4096 个字节），必要时四舍五入。例如:

```
# fdisk /dev/sda
  Command (m for help): p

  Disk /dev/sda: 255 heads, 63 sectors, 1106 cylinders
  Units = cylinders of 16065 * 512 bytes
```

*表 47. Linux 原始 I/O 计算。*

| 设备引导 | 开始 | 结束 | 块 | 标识 | 系统 |
|---|---|---|---|---|---|
| /dev/sda1 | 1 | 523 | 4200997 | 83 | Linux |
| /dev/sda2 | 524 | 1106 | 4682947+ | 5 | 扩展 |
| /dev/sda5 | 524 | 1106 | 4682947 | 83 | Linux |

```
  Command (m for help): q
#
```

/dev/sda5 中的页数是:

```
num_pages = floor( (4682947 * 1024)/4096 )
num_pages = 1170736
```

2. 通过指定磁盘分区名来创建表空间。例如:

```
CREATE TABLESPACE dms1
MANAGED BY DATABASE
USING (DEVICE '/dev/sda5' 1170736)
```

3. 要通过使用联结点（或卷安装点）来指定逻辑分区，将 RAW 分区作为联结点安装到另一个 NTFS 格式的卷上，然后将 NTFS 卷上联结点的路径指定为容器路径。例如:

```
CREATE TABLESPACE TS4
   MANAGED BY DATABASE USING (DEVICE 'C:\JUNCTION\DISK_1' 10000,
      DEVICE 'C:\JUNCTION\DISK_2' 10000)
```

   数据库管理器首先查询分区以了解其中是否存在文件系统 R；如果存在，那么不将该分区视为原始设备，并在该分区上执行一般文件系统 I/O 操作。

数据库管理器所支持的所有其他页大小也支持原始设备上的表空间。

在版本 9 以前，使用 Linux 上的原始控制器实用程序来直接访问磁盘。现在，不推荐使用此方法，也不鼓励使用此方法。如果 Linux 操作系统仍支持此方法，那么数据库管理器就会仍允许使用它，但是，在 db2diag.log 中将有一条消息指示已不推荐使用该方法。

上一种方法要求将磁盘分区与原始控制器“绑定”，然后使用 CREATE TABLESPACE 命令来对数据库管理器指定该原始控制器:

```
CREATE TABLESPACE dms1
MANAGED BY DATABASE
USING (DEVICE '/dev/raw/raw1' 1170736)
```

# 创建表空间

对于非自动存储器表空间，在创建表空间时，必须知道将引用的容器的设备名或文件名。

### 关于本任务

另外，必须知道与要分配给表空间的每个设备名或文件名相关联的空间量。对于自动存储器表空间，数据库管理器将根据与数据库关联的存储路径将容器指定给表空间。

表空间建立数据库系统使用的物理存储设备与用来存储数据的逻辑容器或表之间的关系。

在一个数据库内创建表空间，会将容器分配到表空间，并在数据库系统目录中记录它的定义和属性。然后就可以在此表空间内创建表。创建表空间时，必须知道将引用的容器的设备名或文件名。另外，必须知道与要分配给表空间的每个设备名或文件名相关联的空间量。

当创建数据库时，会创建三个初始表空间。这三个初始表空间的页大小基于使用 CREATE DATABASE 命令时建立或接受的缺省值。此缺省值还表示所有将来 CREATE BUFFERPOOL 和 CREATE TABLESPACE 语句的缺省页大小。如果在创建数据库时不指定页大小，那么缺省页大小是 4 KB。如果在创建表空间时不指定页大小，那么缺省页大小是创建数据库时设置的页大小。

### 过程

- 要使用命令行来创建 SMS 表空间，请输入:

```
CREATE TABLESPACE NAME
            MANAGED BY SYSTEM
      USING ('path')
```

- 要使用命令行来创建 DMS 表空间，请输入:

```
CREATE TABLESPACE NAME
         MANAGED BY DATABASE
      USING (FILE 'path' size)
```

- 要使用命令行来创建自动存储器表空间，请输入下列任一语句:

```
CREATE TABLESPACE NAME
```

```
CREATE TABLESPACE NAME
      MANAGED BY AUTOMATIC STORAGE
```

## 示例

通过使用三个不同的驱动器上的三个目录，下列 SQL 语句在 Windows 上创建了一个 SMS 表空间:

```
CREATE TABLESPACE RESOURCE
           MANAGED BY SYSTEM
     USING ('d:\acc_tbsp', 'e:\acc_tbsp', 'f:\acc_tbsp')
```

以下 SQL 语句使用各自有 5,000 页的两个文件容器创建了一个 DMS 表空间:

```
CREATE TABLESPACE RESOURCE
         MANAGED BY DATABASE
       USING (FILE'd:\db2data\acc_tbsp' 5000,
              FILE'e:\db2data\acc_tbsp' 5000)
```

在前面两个示例中，为容器提供了显式的名称。但是，如果指定相对容器名，那么将在为该数据库创建的子目录中创建容器。

在创建表空间容器时，数据库管理器会创建任何不存在的目录级别。例如，如果将容器指定为 /project/user_data/container1，而目录 /project 不存在，那么数据库管理器会创建目录 /project 和 /project/user_data。

数据库管理器创建的任何目录都是使用 PERMISSION 711 创建的。对于受防护进程访问，PERMISSION 711 是必需的。这意味着实例所有者拥有读写访问权和执行访问权，且其他人拥有执行访问权。任何具有执行访问权的用户还拥有遍历表空间容器目录的权限。因为只有实例所有者具有读写访问权，所以在创建多个实例时，可能会出现下列场景:

- 使用与上面描述的相同的目录结构，假定目录级别 /project/user_data 不存在。
- user1 创建一个实例（缺省情况下命名为 user1），接着创建一个数据库，然后创建一个表空间，且 /project/user_data/container1 作为该表空间的一个容器。
- user2 创建一个实例（缺省情况下命名为 user2），接着创建一个数据库，然后尝试创建一个表空间，且 /project/user_data/container2 作为该表空间的一个容器。

因为数据库管理器根据第一个请求使用 PERMISSION 700 创建了目录级别 /project/user_data，所以 user2 没有对这些目录级别的访问权，因此不能在这些目录中创建 container2。在这种情况下，CREATE TABLESPACE 操作将失败。

解决此冲突有两种方法:

1. 在创建表空间之前创建目录 /project/user_data，并将许可权设置为 user1 和 user2 创建表空间所需的任何访问权。如果所有级别的表空间目录都存在，那么数据库管理器不会修改访问权。
2. 在 user1 创建 /project/user_data/container1 之后，将 /project/user_data 的许可权设置为 user2 创建表空间所需的任何访问权。

如果数据库管理器创建了一个子目录，那么在删除该表空间时数据库管理器也可能将该子目录删除。

在此方案中，假定这些表空间与特定的数据库分区组无关。如果未在该语句中指定下列参数，将使用缺省数据库分区组 IBMDEFAULTGROUP:

```
IN database_partition_group_name
```

通过使用各有 10000 页的三个逻辑卷，下列 SQL 语句在 AIX 系统上创建了一个 DMS 表空间，并指定它们的 I/O 特征:

```
CREATE TABLESPACE RESOURCE
      MANAGED BY DATABASE
      USING (DEVICE '/dev/rdblv6' 10000,
             DEVICE '/dev/rdblv7' 10000,
             DEVICE '/dev/rdblv8' 10000)
      OVERHEAD 7.5
      TRANSFERRATE 0.06
```

在此 SQL 语句中提到的 UNIX 设备必须已经存在，且实例所有者和 SYSADM 组必须能够写入这些设备。

以下示例将在 UNIX 多分区数据库中称为 ODDGROUP 的数据库分区组上创建一个 DMS 表空间。ODDGROUP 必须是先前使用 CREATE DATABASE PARTITION GROUP 语句创建的。在本示例中，假定 ODDGROUP 数据库分区组由编号为 1、3 和 5 的数据库分区组成。在所有数据库分区上都使用具有 10000 个 4 KB 页的 /dev/hdisk0 设备。另外，还为每个数据库分区声明了包含 40000 个 4 KB 页的设备。

```
CREATE TABLESPACE PLANS IN ODDGROUP
       MANAGED BY DATABASE
       USING (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n1hd01' 40000)
             ON DBPARTITIONNUM 1
            (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n3hd03' 40000)
             ON DBPARTITIONNUM 3
            (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n5hd05' 40000)
             ON DBPARTITIONNUM 5
```

通过使用顺序预取工具（它使用并行 I/O），数据库管理器可以极大地提高顺序 I/O 的性能。

您还可以创建一个表空间，它使用的页大小比缺省的 4 KB 大小更大。下列 SQL 语句在 Linux 和 UNIX 系统上创建一个具有 8 KB 页大小的 SMS 表空间。

```
CREATE TABLESPACE SMS8K
      PAGESIZE 8192
      MANAGED BY SYSTEM
      USING ('FSMS_8K_1')
      BUFFERPOOL BUFFPOOL8K
```

注意相关联的缓冲池也必须具有相同的 8 KB 页大小。

只有在激活了创建的表空间所引用的缓冲池之后才能使用该表空间。

可以使用 ALTER TABLESPACE 语句对 DMS 表空间添加、删除容器或调整容器的大小，并修改表空间的 PREFETCHSIZE、OVERHEAD 和 TRANSFERRATE 设置。在执行 ALTER TABLESPACE SQL 语句之后应尽快落实发出表空间语句的事务，以防止发生系统目录争用。

**注:** PREFETCHSIZE 值应该是 EXTENTSIZE 值的倍数。例如，如果 EXTENTSIZE 是 10，那么 PREFETCHSIZE 应为 20 或 30。

还应该考虑通过将 PREFETCHSIZE 设置为 AUTOMATIC 来让数据库管理器自动确定预取大小。

**注:**

直接 I/O（DIO）由于可以绕过在文件系统级别进行高速缓存，从而改进内存性能。此过程可减少 CPU 开销并使得更多的内存可用于数据库实例。

并发 I/O（CIO）具有 DIO 的优点，并且还可以消除串行化写访问权。

DIO 和 CIO 在 AIX 上受支持；DIO 在 HP-UX、Solaris、Linux 和 Windows 操作系统上受支持。

关键字 NO FILE SYSTEM CACHING 和 FILE SYSTEM CACHING 是 CREATE 和 ALTER TABLESPACE SQL 语句的一部分，允许您指定将对每个表空间使用 DIO 还

是 CIO。当 NO FILE SYSTEM CACHING 有效时，只要可能，数据库管理器都会尝试使用“并发 I/O”（CIO）。在不支持 CIO 的情况下（例如，当使用了 JFS 时），将取而使用 DIO。

发出 CREATE TABLESPACE 语句时，缺省情况下将打开已删除的表的恢复功能。此功能使您可使用表空间级的复原和前滚操作来恢复已删除的表数据。这样可比数据库级的恢复要快，且您的数据库将对用户保持可用。

但是，如果有许多删除表操作要恢复或者如果历史记录文件很大，那么正向恢复时已删除的表的恢复功能可能会影响性能。

如果您打算运行许多删除表操作，并且使用循环日志记录或您不想恢复任何已删除的表，那么可能想要禁用此功能。要禁用此功能，可以在发出 CREATE TABLESPACE 语句时显式将 DROPPED TABLE RECOVERY 选项设置为 OFF。此外，可以通过使用 ALTER TABLESPACE 语句关闭现有表空间的已废弃的表的恢复功能。

### 创建系统临时表空间

系统临时表空间用来存储系统临时表。因为系统临时表只能存储在系统临时表空间中，所以数据库必须始终至少有一个这样的表空间。

创建数据库时，定义的三个缺省表空间之一便是名为“TEMPSPACE1”的系统临时表空间。

要创建另一个系统临时表空间，可使用 CREATE TABLESPACE 语句。例如，

```
CREATE SYSTEM TEMPORARY TABLESPACE tmp_tbsp
     MANAGED BY SYSTEM
           USING ('d:\tmp_tbsp','e:\tmp_tbsp')
```

对于每个页大小至少应具有一个表空间。

创建系统临时表空间时，只能指定数据库分区组 IBMTEMPGROUP。

### 创建用户临时表空间

用户临时表空间不是在创建数据库时缺省创建的。如果您的应用程序需要使用临时表，您需要创建临时表将驻留的用户临时表空间。

与常规表空间一样，可在并非 IBMTEMPGROUP 的任何数据库分区组中创建用户临时表空间。创建用户临时表时使用的缺省数据库分区组是 IBMDEFAULTGROUP。

DECLARE GLOBAL TEMPORARY TABLE 语句定义供在用户临时表空间中使用的已声明临时表。

要创建用户临时表空间，可使用 CREATE TABLESPACE 语句:

```
CREATE USER TEMPORARY TABLESPACE usr_tbsp
     MANAGED BY DATABASE
           USING (FILE 'd:\db2data\user_tbsp' 5000,
                  FILE 'e:\db2data\user_tbsp' 5000)
```

## 改变表空间

要使用命令行来改变表空间，可使用 ALTER TABLESPACE 语句。

可以改变 SMS、DMS 和自动存储器容器。还可以重命名表空间，并将它从脱机方式切换至联机方式。

## 改变 SMS 表空间

对于 SMS 表空间，只能将一个容器添加至单一分区数据库，或者将一个或多个容器添加至分区数据库。

执行此操作的过程与『添加或扩展 DMS 容器』中所描述的相同。

## 改变 DMS 表空间

对于 DMS 表空间，可以添加、扩展、重新平衡、删除或减少容器，或者调整容器大小。

### 添加或扩展 DMS 容器

通过将一个或多个容器添加至 DMS 表空间（即，使用 MANAGED BY DATABASE 子句创建的表空间），可以增大该表空间的大小。

当将新容器添加到表空间或扩展现有容器时，可能会发生表空间重新平衡。重新平衡过程涉及将表空间扩展数据块从一个位置移至另一位置。在此过程中，将尝试在表空间内分割数据。重新平衡不必在所有容器上进行，但这取决于许多因素，例如，现有容器配置、新容器的大小和表空间满的程度。

将容器添加到现有表空间时，可能不会从组合分割区 0 开始添加它们，如 第 144 页的『DMS 表空间映射』中所述。在映射中的什么位置开始添加它们是由数据库管理器确定的，并且取决于要添加的容器的大小。如果要添加的容器不够大，那么它的放置位置可能是结束于映射的最后一个组合分割区。如果要添加的容器足够大，那么它的位置会从组合分割区 0 开始。

如果正在添加新的容器且创建新的分割集，那么不会发生重新平衡。新的分割集是在 ALTER TABLESPACE 语句上使用 BEGIN NEW STRIPE SET 子句创建的。还可以在 ALTER TABLESPACE 语句上使用 ADD TO STRIPE SET 子句将容器添加至现有分割集。

在重新平衡期间，不限制对该表空间的访问。如果需要添加多个容器，那么应该同时添加这些容器。

要使用命令行将容器添加到 DMS 表空间，请输入以下内容:

```
ALTER TABLESPACE <name>
   ADD (DEVICE '<path>' <size>, FILE '<filename>' <size>)
```

以下示例说明如何将两个新设备容器（各含 10000 页）添加到 Linux 和 UNIX 系统上的表空间:

```
ALTER TABLESPACE RESOURCE
        ADD (DEVICE '/dev/rhd9'  10000,
                  DEVICE  '/dev/rhd10' 10000)
```

注意，ALTER TABLESPACE 语句允许更改可以影响性能的表空间的其他属性。

### 重新平衡 DMS 容器

ALTER TABLESPACE 语句允许向现有表空间添加容器或扩展容器，以增加其存储容量。

不能手动将容器添加至自动存储器表空间。数据库管理器将在需要时自动扩展或添加容器。

创建表空间时，会创建其表空间映射并对齐所有初始容器，以使它们都从分割区 0 开始。这意味着数据将均匀分布在所有表空间容器上，直到个别容器已满。（请参阅示例 1。）

添加比现有容器小的容器会导致数据分布不均匀。这可能导致并行 I/O 操作（如预取数据）的执行效率比对大小相同的容器执行的效率要低。

向表空间添加新容器或扩展现有容器时，*可能*发生表空间数据的重新平衡。

### 重新平衡

添加或扩展容器时的在平衡过程涉及将表空间扩展数据块从一个位置移动到另一位置，这是通过试图保持数据在表空间内成为分割区来完成的。

在重新平衡期间不会限制对表空间的访问；可以像平常一样删除、创建、填充和查询对象。但是，重新平衡操作可能对性能有很大的影响。如果需要添加多个容器，并且计划重新平衡容器，那么应在单个 ALTER TABLESPACE 语句中同时添加它们，以免数据库管理器不得不多次重新平衡数据。

表空间*高水位标记*在重新平衡过程中起着关键作用。高水位标记是表空间中分配的最高页的页数。例如，表空间有 1000 页，扩展数据块大小为 10，那么结果为 100 个扩展数据块。如果第 42 个扩展数据块是表空间中最高分配的扩展数据块，那么高水位标记是 42 * 10 = 420 页。这与已使用的页不同，因为可能已经释放了高水位标记下的一些扩展数据块，所以它们可供复用。

在重新平衡启动之前，会根据所作的容器更改构建新的表空间映射。重新平衡程序将扩展数据块从由当前映射确定的位置移至由新映射确定的位置。重新平衡程序从扩展数据块 0 开始，一次移动一个扩展数据块，直到移动了持有高水位标记的扩展数据块为止。移动每个扩展数据块时，当前映射每次改变一块以使其看起来与新映射相似。当完成重新平衡时，对于当前映射和新映射，一直到持有高水位标记的分割区，看起来应完全相同。于是使当前映射与新映射看起来完全相同，重新平衡过程就完成了。如果扩展数据块在当前映射中的位置与它在新映射中的位置相同，那么不移动该扩展数据块，并且不发生 I/O。

当添加新容器时，该容器在新映射内的位置取决于其大小及其分割集中其他容器的大小。如果容器足够大，以至于它可以从分割集中的第一个分割区开始，并在分割集中的最后一个分割区处（或以外）结束，那么将它使用该方法放置（请参阅示例 2）。如果容器不够大，无法做到这一点，那么它将在映射中定位为在分割集的最后一个分割区结束（请参阅示例 4。）这样做是为了最小化需要重新平衡的数据量。

**注**：在下列示例中，容器大小未将容器标记计算在内。容器大小很小，仅用于说明目的，它们不是建议的容器大小。这些示例显示表空间中不同大小的容器，但建议使用相同大小的容器。

示例 1:

如果您创建的表空间有三个容器，扩展数据块大小为 10，并且容器分别为 60、40 和 80 页（6、4 和 8 个扩展数据块），那么将创建带有映射的表空间，可进行如第 178 页的

图 13 中所示的图解。

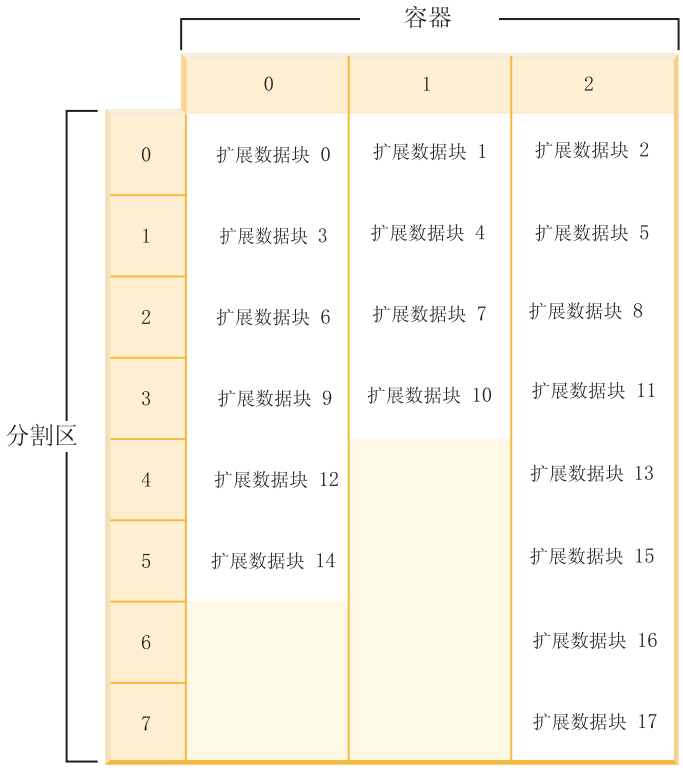



*图 13. 带有三个容器和 18 个扩展数据块的表空间*

相应的表空间映射，如表空间快照中所示，类似如下:

| 范围编号 | 分割集 | 分割区偏移 | 最大扩展数据块 | 最大页 | 起始分割区 | 结束分割区 | 调节 | 容器 |
|---|---|---|---|---|---|---|---|---|
| [0] | [0] | 0 | 11 | 119 | 0 | 3 | 0 | 3（1 和 2） |
| [1] | [0] | 0 | 15 | 159 | 4 | 5 | 0 | 2（0 和 2） |
| [2] | [0] | 0 | 17 | 179 | 6 | 7 | 0 | 1（2） |

表空间映射中的标题是“范围编号”、“分割集”、“分割区偏移”、“根据范围寻址的最大扩展数据块编号”、“根据范围寻址的最大页编号”、“起始分割区”、“结束分割区”、“范围调节”和“容器列表”。

示例 2:

如果在示例 1 中向表空间添加了一个 80 页的容器，容器就大到足以从第一个分割区（分割区 0）中开始，并在最后一个分割区（分割区 7）中结束。它被定位为从第一个分割区中开始。结果表空间可进行如第 179 页的图 14 中所示的图解。

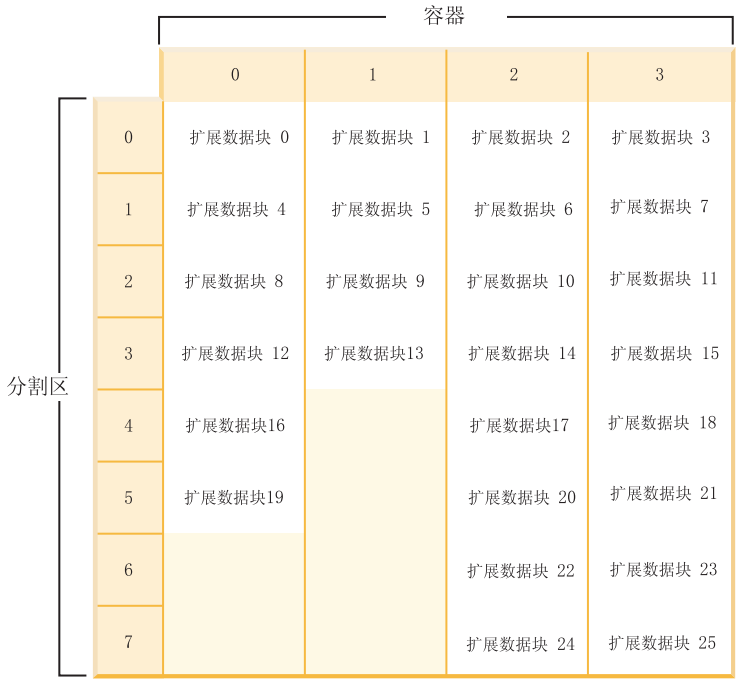



*图 14. 带有四个容器和 26 个扩展数据块的表空间*

相应的表空间映射，如表空间快照中所示，类似如下:

| 范围编号 | 分割集 | 分割区偏移 | 最大扩展数据块 | 最大页 | 起始分割区 | 结束分割区 | 调节 | 容器 |
|---|---|---|---|---|---|---|---|---|
| [0] | [0] | 0 | 15 | 159 | 0 | 3 | 0 | 4（0、1、2 和 3） |
| [1] | [0] | 0 | 21 | 219 | 4 | 5 | 0 | 3（0、2 和 3） |
| [2] | [0] | 0 | 25 | 259 | 6 | 7 | 0 | 2（2 和 3） |

如果高水位标记在扩展数据块 14 以内，那么重新平衡程序将从扩展数据块 0 开始，并且将把所有扩展数据块上移至 14（包括 14）。两个映射内的扩展数据块 0 的位置相同，所以不必移动此扩展数据块。扩展数据块 1 和 2 的情况相同。需要移动扩展数据块 3，所以从旧位置（容器 0 内的第二个扩展数据块）读取该扩展数据块并写至新位置（容器 3 内的第一个扩展数据块）。将移动此扩展数据块之后直到扩展数据块 14（包括扩展数据块 14）的每个扩展数据块。一旦移动了扩展数据块 14，当前映射看起来会像新映射，并且重新平衡程序将终止。

如果改变映射以使所有新添加的空间都在高水位标记之后，那么不需要重新平衡并且所有的空间都立即可用。如果改变映射以使部分空间在高水位标记之后，那么分割区中在高水位标记之上的空间将是可用的。余下部分直到重新平衡完成才可用。

如果决定扩展容器，那么重新平衡程序的功能相似。如果扩展容器以使它超出了分割集中的最后一个分割，那么将扩展该分割集以适应这种情况并将相应地移出其后的分割集。结果是容器不会扩展到其后的任何分割集中。

示例 3:

以“示例 1”中的表空间为例。如果将容器 1 从 40 页扩展到 80 页，那么新表空间将类似图 15。

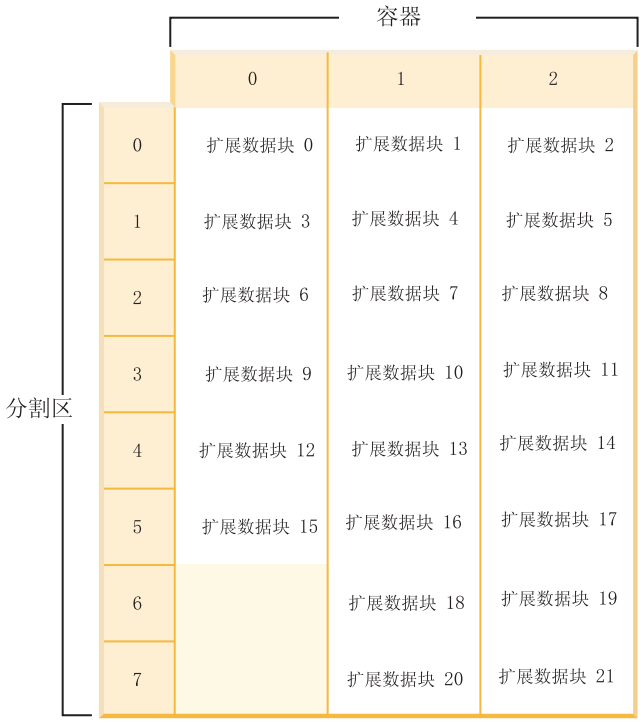



*图 15. 带有三个容器和 22 个扩展数据块的表空间*

相应的表空间映射，如表空间快照中所示，类似如下:

```
范围      分割      分割区      最大      最大      起始      结束      调节    容器
编号      集        偏移   扩展数据块   页       分割区    分割区
   [0]    [0]       0          17        179       0         5         0    3（0、1 和 2）
   [1]    [0]       0          21        219       6         7         0    2（1 和 2）
```

示例 4:

请考虑示例 1 中的表空间。如果向它添加一个 50 页（5 个扩展数据块）的容器，那么将以如下方式将该容器添加至新映射。容器大小不足以从第一个分割区（分割区 0）中开始，并在最后一个分割区（分割区 7）处或以外结束，因此将它定位为在最后一个分割区中结束。（请参阅第 181 页的图 16。）

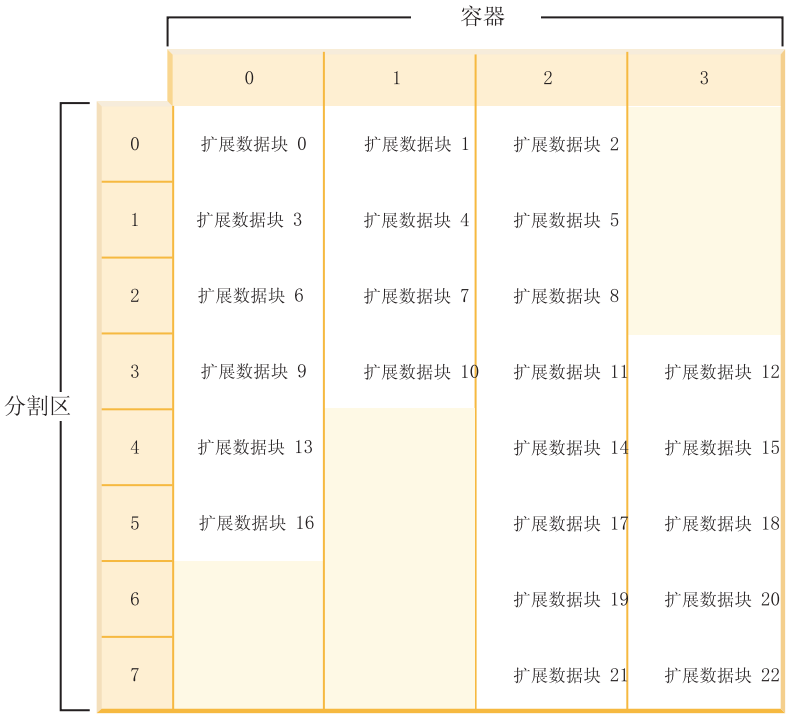



*图 16. 带有四个容器和 23 个扩展数据块的表空间*

相应的表空间映射，如表空间快照中所示，类似如下：

| 范围编号 | 分割集 | 分割区偏移 | 最大扩展数据块 | 最大页 | 起始分割区 | 结束分割区 | 调节 | 容器 |
|---|---|---|---|---|---|---|---|---|
| [0] | [0] | 0 | 8 | 89 | 0 | 2 | 0 | 3（0、1 和 2） |
| [1] | [0] | 0 | 12 | 129 | 3 | 3 | 0 | 4（0、1、2 和 3） |
| [2] | [0] | 0 | 18 | 189 | 4 | 5 | 0 | 3（0、2 和 3） |
| [3] | [0] | 0 | 22 | 229 | 6 | 7 | 0 | 2（2 和 3） |

要扩展容器，在 ALTER TABLESPACE 语句上使用 EXTEND 或 RESIZE 子句。要添加容器并重新平衡数据，在 ALTER TABLESPACE 语句上使用 ADD 子句。如果正在向已经有多个分割集的表空间添加容器，那么可以指定想要向哪个分割集添加容器。为此，在 ALTER TABLESPACE 语句上使用 ADD TO STRIPE SET 子句。如果不指定分割集，那么缺省行为将是向当前分割集添加容器。当前分割集是最新创建的分割集，而不是最后向其添加空间的分割集。

对分割集的任何更改可能导致对该分割集及其后的任何其他分割集的重新平衡。

可以通过使用表空间快照来监视重新平衡的进度。表空间快照可以提供关于重新平衡的信息，如重新平衡的开始时间、已经移动了多少个扩展数据块以及需要移动多少个扩展数据块。

## 不重新平衡（使用分割集）

如果添加或扩展容器，并且添加的空间在表空间高水位标记之上，那么不会发生重新平衡。

添加容器将始终在高水位标记下添加空间。换句话说，添加容器时，重新平衡通常是必要的。有一个选项可强制将新容器添加到高水位标记之上，它允许您选择不对表空

间的内容重新平衡。此方法的一个优点是新容器可立即使用。不进行重新平衡这一选项仅在添加容器时才适用，在扩展现有容器时不适用。扩展容器时，仅当添加的空间在高水位标记之上时，才能避免重新平衡。例如，如果有许多大小相同的容器，并且按相同的量扩展它们，那么扩展数据块的相对位置不会更改，并且不会发生重新平衡。

将容器添加到表空间中而不进行重新平衡可通过添加新的*分割集*来实现。分割集是表空间中的一组容器，数据在其上进行分割，且独立于属于该表空间的其他容器。现有分割集中的现有容器保持不变，而添加的容器成为新分割集的一部分。

要添加容器而不进行重新平衡，在 ALTER TABLESPACE 语句上使用 BEGIN NEW STRIPE SET 子句。

示例 5:

如果表空间有三个容器，扩展数据块大小为 10，并且容器分别为 30、40 和 40 页（分别为 3、4 和 4 个扩展数据块），那么表空间可进行如图 17 中所示的图解。

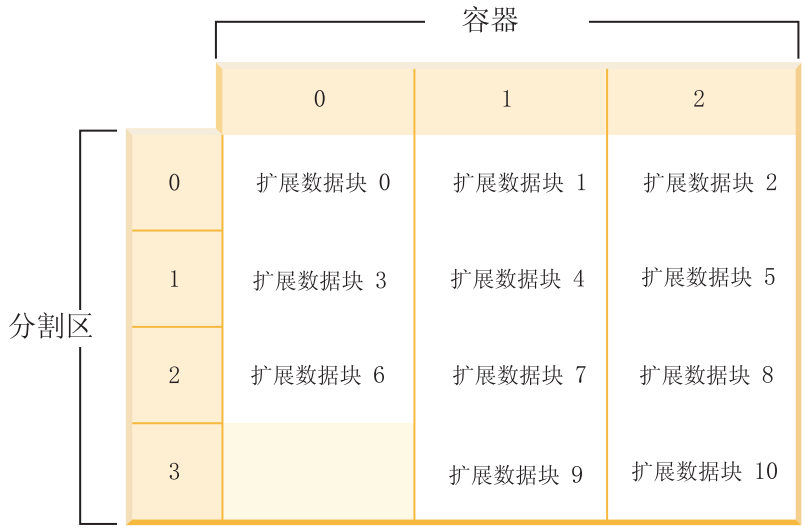



*图 17. 带有三个容器和 11 个扩展数据块的表空间*

相应的表空间映射，如表空间快照中所示，类似如下:

```
范围      分割     分割区     最大     最大      起始      结束     调节    容器
编号       集       偏移  扩展数据块  页       分割区    分割区
   [0]    [0]       0         8          89       0         2         0     3（0、1 和 2）
   [1]    [0]       0         10         109      3         3         0    2（1 和 2）
```

示例 6:

在使用 BEGIN NEW STRIPE SET 子句添加 30 页和 40 页的两个新容器（分别为 3 和 4 个扩展数据块）时，不会影响现有范围；而是将创建一组新范围。这一组新范围是一个分割集，而最新创建的分割集称为当前分割集。添加了两个新的容器之后，表空

间将类似图 18。

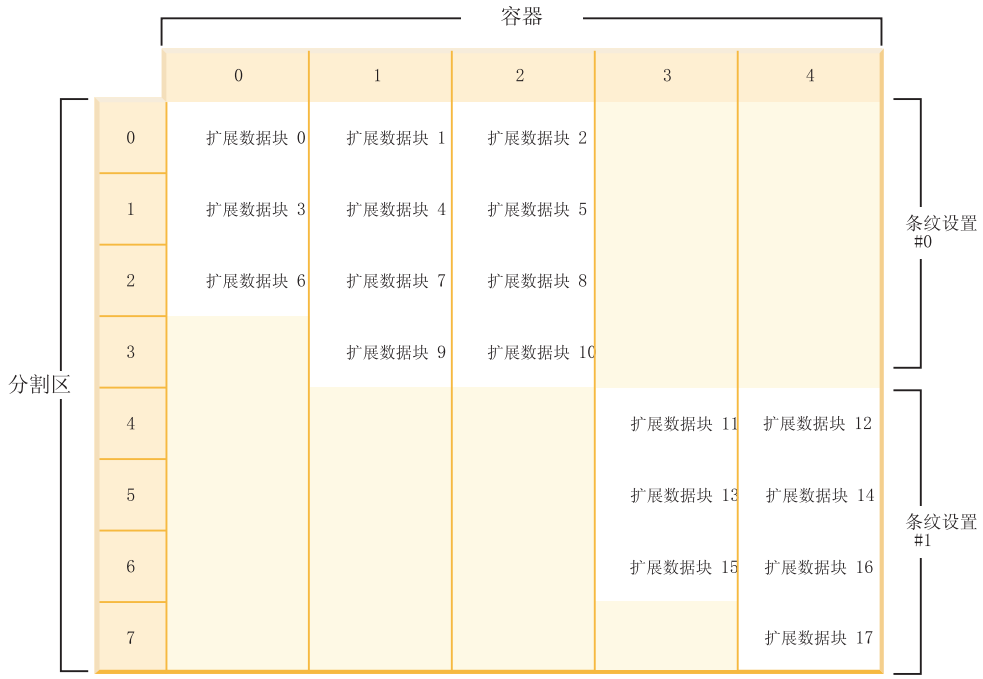



*图 18. 带有两个分割集的表空间*

相应的表空间映射，如表空间快照中所示，类似如下:

| 范围编号 | 分割集 | 分割区偏移 | 最大扩展数据块 | 最大页 | 起始分割区 | 结束分割区 | 调节 | 容器 |
|---|---|---|---|---|---|---|---|---|
| [0] | [0] | 0 | 8 | 89 | 0 | 2 | 0 | 3（0、1 和 2） |
| [1] | [0] | 0 | 10 | 109 | 3 | 3 | 0 | 2（1 和 2） |
| [2] | [1] | 4 | 16 | 169 | 4 | 6 | 0 | 2（3 和 4） |
| [3] | [1] | 4 | 17 | 179 | 7 | 7 | 0 | 1（4） |

如果向表空间添加新的容器，并且未将 TO STRIPE SET 子句与 ADD 子句配合使用，那么将把容器添加至当前分割集（最高分割集）。可以使用 ADD TO STRIPE SET 子句来将容器添加到表空间中的任何分割集。必须指定有效的分割集。

数据库管理器使用表空间映射来跟踪分割集，并且添加新容器而不进行重新平衡通常将导致映射比对容器进行重新平衡时增长得快。当表空间映射变得过大时，如果试图添加更多容器，将接收到错误 SQL0259N。

## 调整 DMS 容器的大小

不能手动调用自动存储器表空间中容器的大小。

数据库管理器将在需要时自动扩展或添加容器。但是，可以调整 DMS 表空间（即，使用 MANAGED BY DATABASE 子句创建的表空间）中容器的大小。

只能将每个原始设备用作一个容器。创建了原始设备之后，其大小是固定的。当您考虑使用调整大小或扩展选项来增大原始设备容器时，应先检查原始设备大小以确保您并未试图将设备容器大小增大到大于原始设备大小。

还可以从 DMS 表空间中删除现有容器，减少 DAS 表空间中现有容器的大小以及将新容器添加至 DMS 表空间而不需要在所在容器间重新平衡数据。

仅当正在删除或缩小其大小的扩展数据块数目小于或等于表空间中“高水位标记”之上的可用数据块数目时，才允许删除现有表空间容器以及缩小现有容器的大小。高水位标记是表空间中分配的最高页的页数。此标记与表空间中已使用的页的数目不同，原因是高水位标记下的某些扩展数据块可能可供复用。

表空间中高水位标记之上的可用扩展数据块数非常重要，原因是直至高水位标记（包括高水位标记）的所有扩展数据块必须位于表空间内的同一逻辑位置。结果表空间必须有足够的空间才能容纳所有数据。如果没有足够的可用空间，那么会产生一条错误消息（SQL20170N 或 SQLSTATE 57059）。

要删除容器，可在 ALTER TABLESPACE 语句上使用 DROP 选项。例如:

```
ALTER TABLESPACE TS1 DROP (FILE 'file1', DEVICE '/dev/rdisk1')
```

要缩小现有容器的大小，可使用 RESIZE 选项或 REDUCE 选项。使用 RESIZE 选项时，作为语句的一部分列示的所有容器都必须增大大小或减小大小。不能在同一语句中增大某些容器而缩小其他容器。如果知道容器大小的新下限，应考虑调整大小方法。如果不知道（或不关心）容器的当前大小，那么应该考虑缩小方法。

要使用命令行来缩小 DMS 表空间中一个或多个容器的大小，请输入:

```
ALTER TABLESPACE <name>
   REDUCE (FILE '<filename>' <size>)
```

以下示例说明如何在基于 Windows 的系统上的表空间中缩小文件容器（含 1000 页）:

```
  ALTER TABLESPACE PAYROLL
REDUCE (FILE 'd:\hldr\finance'  200)
```

在此操作之后，文件大小就从 1000 页减少至 800 页。

要使用命令行来增大 DMS 表空间中一个或多个容器的大小，请输入:

```
ALTER TABLESPACE <name>
         RESIZE (DEVICE '<path>' <size>)
```

以下示例说明如何在 Linux 和 UNIX 系统上的表空间中增大两个设备容器（各含 1000 页）:

```
   ALTER TABLESPACE HISTORY
       RESIZE (DEVICE '/dev/rhd7'  2000,
                      DEVICE  '/dev/rhd8' 2000)
```

在此操作之后，两个设备的大小都从 1000 页增加至 2000 页。可在容器间重新平衡该表空间的内容。在重新平衡期间，不限制对该表空间的访问。

要使用命令行来扩展 DMS 表空间中一个或多个容器，请输入:

```
ALTER TABLESPACE <name>
         EXTEND (FILE '<filename>' <size>)
```

以下示例说明如何在基于 Windows 的系统上的表空间中增大文件容器（各含 1000 页）：

```
 ALTER TABLESPACE PERSNEL
EXTEND (FILE 'e:\wrkhist1'  200
        FILE  'f:\wrkhist2'  200)
```

在此操作之后，两个文件的大小都从 1000 页增大至 1200 页。可在容器间重新平衡该表空间的内容。在重新平衡期间，不限制对该表空间的访问。

通过预取程序以并行方式添加或修改 DMS 容器（文件容器和原始设备容器）。要增加这些创建或调整容器大小操作的并行性，可以增加系统中运行的预取程序的数目。不以并行方式执行的唯一进程是记录这些操作以及在创建容器的情况下标记这些容器。

**注**: 要使 CREATE TABLESPACE 或 ALTER TABLESPACE 语句的并行性最大（对于将新的容器添加至现有的表空间），确保预取程序数大于或等于要添加的容器数。预取程序数目由 *num_ioservers* 数据库配置参数控制。必须停止数据库以使新参数值生效。也就是说，必须断开所有应用程序和用户与数据库的连接以使更改生效。

注意，ALTER TABLESPACE 语句允许更改可以影响性能的表空间的其他属性。

## 删除或减少 DMS 容器

对于 DMS 表空间，可以使用 ALTER TABLESPACE 语句从表空间中删除容器或缩小容器的大小。

仅当该操作删除的扩展数据块的数目小于或等于表空间中的高水位标记之上的可用扩展数据块的数目时，才允许删除或缩小容器。这是必须的，因为该操作不能更改页号，从而直到高水位标记（包括高水位标记）的所有扩展数据块在表空间内必须处于相同的逻辑位置。因此，结果表空间必须具有足够的空间才能存放直到高水位标记（包括高水位标记）的所有数据。在没有足够的可用空间的情况下，执行语句时会立即接收到错误。

高水位标记是表空间中分配的最高页的页数。例如，表空间有 1000 页，扩展数据块大小为 10，那么结果为 100 个扩展数据块。如果第 42 个扩展数据块是表空间中最高分配的扩展数据块，那么意味着高水位标记是 42 * 10 = 420 页。这与已使用的页不同，因为可能已经释放了高水位标记下的一些扩展数据块，所以它们可供复用。

删除或缩小容器时，如果数据位于正从表空间中删除的空间中，那么将进行重新平衡。在重新平衡启动之前，会根据所作的容器更改构建新的表空间映射。重新平衡程序将把扩展数据块从由当前映射确定的位置移至由新映射确定的位置。重新平衡程序从包含高水位标记的扩展数据块开始，一次移动一个扩展数据块，直到移动了扩展数据块 0 为止。移动每个扩展数据块时，当前映射每次改变一块以使其看起来与新映射相似。如果扩展数据块在当前映射中的位置与它在新映射中的位置相同，那么不移动该扩展数据块，并且不发生 I/O。因为重新平衡从表空间中分配的最高扩展数据块开始移动，并以表空间中的第一个扩展数据块结束，所以又称为*逆向重新平衡*（与在添加或扩展容器之后向表空间添加空间时发生的*正向重新平衡*相反）。

删除容器时，余下的容器将重新编号，它们的容器标识从 0 开始每次加 1。如果删除了分割集中的所有容器，那么将从映射除去该分割集，并且会下移映射中其后的所有分割集并对其重新编号，以便分割集号码中没有间隔。

**注:** 在下列示例中，容器大小未将容器标记计算在内。容器大小很小，仅用于说明目的，它们不是建议的容器大小。这些示例显示表空间中不同大小的容器，但是这只是用于说明目的；建议使用相同大小的容器。

例如，考虑这样一个表空间，它有三个容器，扩展数据块大小为 10。容器分别为 20、50 和 50 页（2、5 和 5 个扩展数据块）。表空间的图表显示在图 19 中。

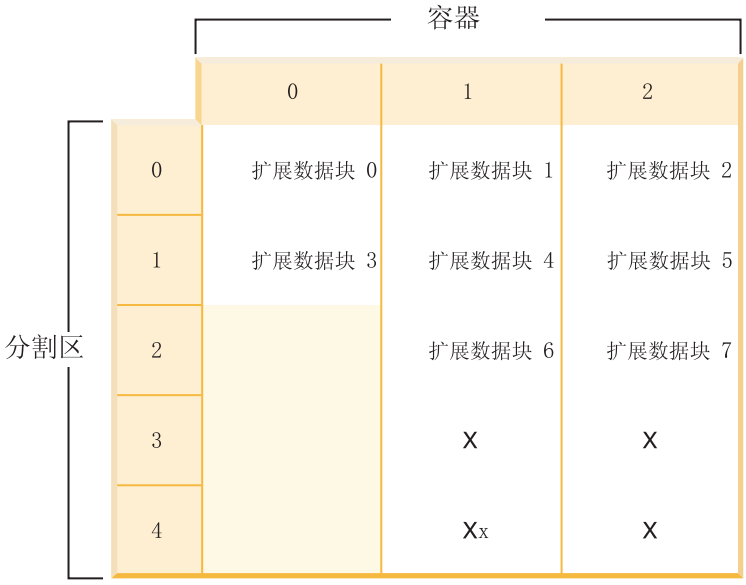



*图 19. 带有 12 个扩展数据块（包括四个无数据的扩展数据块）的表空间*

X 指示存在扩展数据块，但其中没有数据。

如果想要删除有两个扩展数据块的容器 0，那么高水位标记之上必须至少具有两个可用扩展数据块。高水位标记在扩展数据块 7 中，有四个可用扩展数据块，因此可以删除容器 0。

相应的表空间映射，如表空间快照中所示，类似如下:

```
范围     分割     分割区    最大      最大      起始      结束      调节    容器
编号       集      偏移   扩展数据块   页      分割区    分割区
  [0]     [0]       0         5       59        0         1        0      3
                                                                          （0、1 和 2）
  [1]     [0]       0        11      119        2         4        0      2
                                                                          （1 和 2）
```

删除之后，表空间将仅有容器 0 和容器 1。新的表空间图表显示在第 187 页的图 20 中。

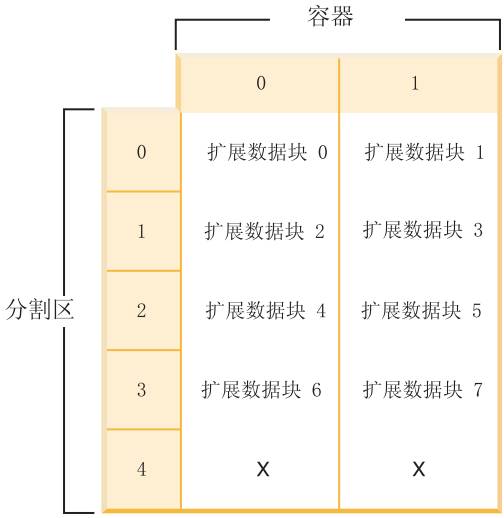



*图 20. 删除容器之后的表空间*

相应的表空间映射，如表空间快照中所示，类似如下:

```
范围      分割      分割区     最大      最大      起始      结束      调节     容器
编号       集        偏移   扩展数据块   页       分割区    分割区
  [0]     [0]        0         9        99         0         4         0      2
                                                                            (0 和 1)
```

如果想要缩小容器，重新平衡程序以相似的方式工作。

要缩小容器，在 ALTER TABLESPACE 语句上使用 REDUCE 或 RESIZE 选项。要删除容器，在 ALTER TABLESPACE 语句上使用 DROP 选项。

## 改变自动存储器表空间

对于自动存储器表空间，只能减小容器大小。

执行此操作的过程与第 183 页的『调整 DMS 容器的大小』中所描述的相同。

# 重命名表空间

使用 RENAME TABLESPACE 语句来重命名表空间。

不能重命名 SYSCATSPACE 表空间。不能重命名处于“前滚暂挂”或“正在前滚”状态的表空间。

当复原在备份后已被重命名的表空间时，必须在 RESTORE DATABASE 命令中使用新的表空间名。如果使用先前的表空间名，那么将找不到该名称。同样，如果使用 ROLLFORWARD DATABASE 命令前滚该表空间，也需确保使用新名称。如果使用先前的表空间名，那么将找不到该名称。

可以给予现有表空间新名称，而无需关心该表空间中的个别对象。重命名表空间时，将更改所有引用该表空间的目录记录。

## 将表空间从脱机状态切换至联机状态

如果与表空间相关的容器已变得可访问，可以使用 ALTER TABLESPACE 语句的 SWITCH ONLINE 子句从表空间中除去 OFFLINE 状态。

表空间已除去 OFFLINE 状态，而数据库的其余部分仍在运行并在使用中。

使用此子句的一个替代方法是将所有应用程序与数据库断开连接，然后再次将这些应用程序连接至数据库。这样就可以从表空间中除去 OFFLINE 状态。

要使用命令行从表空间中除去 OFFLINE 状态，请输入:

```
db2 ALTER TABLESPACE <name>
      SWITCH ONLINE
```

## 当数据位于 RAID 设备上时优化表空间性能

要在将数据存放在“独立磁盘冗余阵列”（RAID）设备中时优化性能，请遵循下列准则。

1. 在一组 RAID 设备上创建表空间时，应该为不同设备上的给定表空间（SMS 或 DMS）创建容器。

   考虑以下示例: 您将 15 个 146GB 磁盘配置为三个 RAID-5 阵列，每个阵列包含 5 个磁盘。格式化以后，每个磁盘可容纳大约 136GB 数据。因此，每个阵列可存储大约 544GB（4 个活动磁盘 x 136GB）数据。如果表空间需要 300GB 存储空间，那么创建三个容器并将每个容器放在不同的设备上。每个容器使用设备上的 100GB（300GB/3），并且每个设备上保留 444GB（544GB - 100GB）以提供附加表空间。
2. 为表空间选择适当的扩展数据块大小。表空间的扩展数据块大小表示数据库管理器向当前容器写入多少数据才转向下一个容器。理想状态下，扩展数据块大小应该是磁盘底层段大小的倍数，其中段大小表示磁盘控制器向一个物理磁盘写入多少数据才转向下一个物理磁盘。选择应该是段大小倍数的扩展数据块大小，以确保基于扩展数据块的操作（如预取时的并行顺序读取）不会争用相同的物理磁盘。同时选择应该是页大小倍数的扩展数据块大小。

   在本示例中，如果段大小为 64KB，而页大小为 16KB，那么适当的扩展数据块大小可能是 256KB。
3. 使用 DB2_PARALLEL_IO 注册表变量来对所有表空间启用并行 I/O，并对每个容器指定物理磁盘数。

   对于此示例中的情况，请将 DB2_PARALLEL_IO 设置为 *:4。

   如果将表空间的预取大小设置为 AUTOMATIC，那么数据库管理器将使用您对 DB2_PARALLEL_IO 指定的物理磁盘数值来确定预取大小值。如果预取大小未设置为 AUTOMATIC，那么您可以手动设置此值，请考虑 RAID 条带大小，它是段大小乘以活动磁盘数产生的值。考虑满足下列条目的预取大小值:

   - 它等于 RAID 条带大小乘以 RAID 并行设备数（或此乘积的整数表示）。

- 它是扩展数据块大小的整数表示。

在本示例中，可将预取大小设置为 768KB。此值等于 RAID 条带大小（256KB）乘以 RAID 并行设备数（3）。它也是扩展数据块大小（256KB）的倍数。选择此预取大小意味着单个预取会涉及所有阵列中的所有磁盘。如果因为工作负载主要涉及大量扫描而希望预取程序更积极地工作，那么可改为使用此值的倍数，如 1536 KB（768 KB x 2）。

4. 不要设置 DB2_USE_PAGE_CONTAINER_TAG 注册表变量。如之前所述，应使用等于 RAID 条带大小或其倍数的扩展数据块大小来创建表空间。但是，将 DB2_USE_PAGE_CONTAINER_TAG 设置为 ON 时，将使用单页容器标记，并且扩展数据块不会与 RAID 条带对齐。因此，在 I/O 请求期间可能需要访问比最优情况更多的物理磁盘。

## 删除表空间

当删除表空间时，也会删除该表空间中的所有数据，释放容器，除去目录条目，并导致该表空间中定义的所有对象都被删除或标记为无效。

可以通过删除表空间来复用空表空间中的容器，但是，在试图复用这些容器之前，必须落实该 DROP TABLESPACE 语句。

**注**: 如果不删除与某个表空间相关联的所有表空间，那么不能删除该表空间。例如，如果您在一个表空间中创建了表并在另一个表空间中创建了其索引，那么需要在一个 DROP TABLESPACE 命令中同时删除索引和数据表空间。

### 删除用户表空间

可删除一个包含所有表数据的用户表空间，包括在该单个用户表空间中的索引和 LOB 数据。也可删除所包含的表跨几个表空间的用户表空间。即，可能表数据在一个表空间，索引在另一个表空间且任何 LOB 在第三个表空间。必须在一条语句中同时删除所有三个表空间。包含跨越的表的所有表空间必须全部纳入此单条语句中，否则该删除请求将失败。

要使用命令行来删除用户表空间，请输入:

```
DROP TABLESPACE <name>
```

以下 SQL 语句将删除表空间 ACCOUNTING:

```
DROP TABLESPACE ACCOUNTING
```

### 删除用户临时表空间

仅当用户临时表空间中当前未定义已声明临时表时，才能删除该表空间。当删除表空间时，不会尝试删除该表空间中的所有已声明临时表。

**注**: 已声明临时表是在说明它的应用程序与数据库断开连接时隐式删除的。

### 删除系统临时表空间

如果不首先创建另一系统临时表空间，那么不能删除页大小为 4 KB 的系统临时表空间。新的系统临时表空间必须具有 4 KB 页大小，原因是数据库必须始终存在至少一个具有 4 KB 页大小的系统临时表空间。例如，如果具有页大小为 4 KB 的单个系统临时表空间，并且您想要将一个容器添加到该表空间（它是 SMS 表空间），那么您必须首先添加一个具有适当数目的容器的新 4 KB 页大小的系统临时表空间，然后删除旧的系统临时表空间。（如果正在使用 DMS，那么可以添加容器而不必删除并重新创建表空间。）

缺省表空间页大小是创建数据库时使用的页大小（缺省情况下为 4 KB，但也可以为 8 KB、16 KB 或 32 KB）。

这是用来创建系统临时表空间的语句:

```
CREATE SYSTEM TEMPORARY TABLESPACE <name>
     MANAGED BY SYSTEM USING ('<directories>')
```

之后，要使用命令行删除系统表空间，请输入以下内容:

```
DROP TABLESPACE <name>
```

以下 SQL 语句创建一个称为 TEMPSPACE2 的新的系统临时表空间:

```
CREATE SYSTEM TEMPORARY TABLESPACE TEMPSPACE2
        MANAGED BY SYSTEM USING ('d:\systemp2')
```

一旦创建了 TEMPSPACE2，那么可使用以下命令删除原来的系统临时表空间 TEMPSPACE1:

```
DROP TABLESPACE TEMPSPACE1
```

# 第 10 章 模式

模式是已命名对象的集合；它提供一种方法来按逻辑分组这些对象。模式也是名称限定词；它提供一种方法来对几个对象使用相同自然名称，并防止对这些对象进行二义性引用。

例如，使用模式名“INTERNAL”和“EXTERNAL”很容易区分两个不同的 SALES 表（INTERNAL.SALES 和 EXTERNAL.SALES）。

模式还允许多个应用程序将数据存储在单个数据库中，而不会遇到名称空间冲突。

模式与 *XML 模式*不同，不应将它们混淆。XML 模式是一种描述 XML 文档的结构并验证其内容的标准。

模式可以包含表、视图、昵称、触发器、函数、程序包和其他对象。模式本身是一个数据库对象。可使用 CREATE SCHEMA 语句显式创建模式，并且将当前用户或指定的授权标识记录为模式所有者。如果用户具有 IMPLICIT_SCHEMA 权限，那么也可以在创建另一个对象时隐式创建模式。

*模式名*用作两部分对象名的前半部分。如果在创建对象时使用模式名专门限定了该对象，那么该对象被指定给该模式。如果在创建对象时未指定模式名，那么将使用缺省模式名（在 CURRENT SCHEMA 专用寄存器中指定）。

例如，具有 DBADM 权限的用户为用户 A 创建模式 C:

```
CREATE SCHEMA C AUTHORIZATION A
```

然后，用户 A 可以发出以下语句在模式 C 中创建表 X（前提是该用户 A 具有 CREATETAB 数据库权限）：

```
CREATE TABLE C.X (COL1 INT)
```

某些模式名是保留名。例如，内置函数属于 SYSIBM 模式，而预先安装的用户定义的函数属于 SYSFUN 模式。

如果在创建数据库时未使用 RESTRICTIVE 选项，那么所有用户将具有 IMPLICIT_SCHEMA 权限。只要具有此权限，无论用户何时使用不存在的模式名创建对象，都会隐式创建一个模式。隐式创建模式时，将授予 CREATEIN 特权，该特权允许任何用户在此模式中创建其他对象。创建这些对象（如别名、单值类型、函数和触发器）的能力扩展为隐式创建模式。对隐式创建的模式的缺省特权提供了与先前版本的向后兼容性。

如果撤销 PUBLIC 的 IMPLICIT_SCHEMA 权限，那么可以使用 CREATE SCHEMA 语句显式创建模式，或者由授予了 IMPLICIT_SCHEMA 权限的用户（例如，具有 DBADM 权限的用户）隐式创建。虽然撤销 PUBLIC 的 IMPLICIT_SCHEMA 权限会加大对模式名使用的控制，但在现有应用程序尝试创建对象时它会导致权限错误。

模式也具有特权，它们允许模式所有者控制哪些用户有权创建、改变、复制和删除模式中的对象。这提供了一种方法来控制对数据库中的对象子集的处理。模式所有者最初被授予对该模式的所有这些特权，并且他们能够将特权授予给其他人。隐式创建的

© Copyright IBM Corp. 1993, 2010

模式由系统拥有，并且所有用户最初被授予在这种模式中创建对象的特权。具有 SYSADM 或 DBADM 权限的用户可以更改用户拥有的对任何模式的特权。因此，可以控制在任何模式（即使是隐式创建的模式）中创建、改变、复制和删除对象的访问权。

## 设计模式

将数据组织成表时，将表和其他相关对象分组在一起可能会有好处。为此，须使用 CREATE SCHEMA 语句来定义一个模式。

有关该模式的信息保存在连接的数据库的系统目录表中。在创建其他对象时，可以将它们置于您创建的模式内，但要注意，一个对象只能存在于一个模式中。

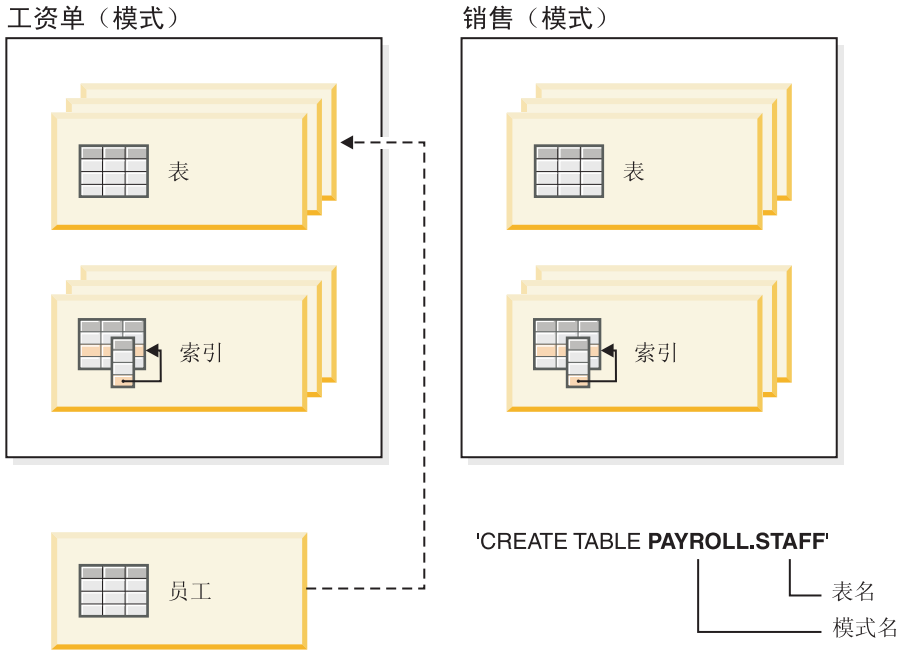
可以将模式比作目录，并且当前模式相当于当前目录。通过这种类比，SET SCHEMA 相当于 change directory 命令。

**要点**：了解除下面描述的缺省 CURRENT SCHEMA 设置除外，授权标识与模式之间没有关联这一点很重要。

在设计数据库和表时，还应考虑系统中的模式，包括模式名和将与每个模式关联的对象。

为数据库中的大多数对象指定一个由两部分组成的唯一名称。第一（最左边的）部分称为限定词或模式，而第二（最右边的）部分称为简单（或未限定）名称。从句法上来说，这两部分并置成用句点分隔的单个字符串。第一次创建可以由模式名限定的任何对象（例如，表、索引、视图、用户定义的数据类型、用户定义的函数、昵称、程序包或触发器）时，会根据对象名称中的限定词将该对象指定给一个特定模式。

例如，下图说明在创建表的过程中如何将该表指定给一个特定模式：




您还应该了解如何授予模式访问权，以便为用户授予正确的权限并提供指示信息:

**模式名** 在创建新模式时，名称不能标识目录中已描述的模式名，并且不能以“SYS”开头。有关其他限制和建议，请参阅第 195 页的『模式名限制和建议』。

**访问模式**

由于使用模式来强制数据库中的唯一性，所以不允许对模式内的对象的非限定访问。当考虑两个用户有可能使用同一名称创建两个表（或其他对象）时，这一点变得很清楚。没有模式来强制唯一性时，若第三个用户试图查询该表，将会造成不明确。没有进一步的限定时，不可能确定要使用哪一个表。

作为 CREATE SCHEMA 语句的一部分创建的任何对象的定义者是模式所有者。此所有者可以授予和撤销其他用户的模式特权。

如果用户具有 SYSADM 或 DBADM 权限，那么该用户可以使用任何有效名称来创建模式。当创建数据库时，会将 IMPLICIT_SCHEMA 权限授予 PUBLIC（即，授予所有用户）。

如果用户没有 IMPLICIT_SCHEMA 或 DBADM 权限，那么他们可以创建的唯一模式是具有与他们自己的授权标识同名的模式。

**缺省模式**

如果未将模式或限定符指定为要创建的对象名的一部分，那么该对象将指定给 CURRENT SCHEMA 专用寄存器中所指示的缺省模式。此专用寄存器的缺省值是会话授权标识的值。

动态语句中的未限定对象引用需要缺省模式。通过将 CURRENT SCHEMA 专用寄存器设置为要用作缺省模式的模式，可以设置特定 DB2 连接的缺省模式。不需要指定的权限来设置此专用寄存器，因此任何用户都可以设置 CURRENT SCHEMA。

SET SCHEMA 语句的语法如下:

```
SET SCHEMA = <schema-name>
```

可采用交互方式或从应用程序中发出此语句。CURRENT SCHEMA 专用寄存器的初始值等于当前会话用户的授权标识。有关更多信息，请参阅 SET SCHEMA 语句。

**注:**

- 可使用其他方法来在连接时设置缺省模式。例如，通过对 CLI/ODBC 应用程序使用 `cli.ini` 文件，或者通过对 JDBC 应用程序编程接口使用连接属性。
- 缺省模式记录不是在系统目录中创建的，但只要未将模式或限定符指定为要创建的对象名的一部分，该记录就只作为数据库管理器可以从 CURRENT SCHEMA 专用寄存器获取的值存在。

**隐式创建**

如果您具有 IMPLICIT_SCHEMA 权限，那么可以隐式创建模式。只要具有此权限，无论您何时使用不存在的模式名创建对象，都会隐式创建一个模式。只要创建对象的用户拥有 IMPLICIT_SCHEMA 权限，通常会在第一次创建模式中的数据对象时隐式创建模式。

**显式创建**

还可以通过在命令行或应用程序中执行 CREATE SCHEMA 和 DROP SCHEMA 语句来显式创建和删除模式。有关更多信息，请参阅 CREATE SCHEMA 和 DROP SCHEMA 语句。

**按模式分组的表和视图别名**

要允许另一个用户访问表或视图而不必输入模式名作为表名或视图名的限定部分，需要为该用户建立别名。别名定义将定义包括用户模式的标准表名或视图名；然后用户将只需要使用别名进行查询。将通过作为别名定义一部分的用户模式对别名进行全限定。

## 根据模式将对象分组

数据库对象名可由单个标识组成，也可以是由两个标识组成的*模式限定对象*。模式限定对象的模式或高位部分提供了一种将数据库中的对象分类或分组的方法。当创建如表、视图、别名、单值类型、函数、索引、程序包或触发器之类的对象时，会给它分配一个模式。此赋值可显式或隐式地执行。

在一条语句中引用对象时，如果使用了两部分对象名的高位部分，那么显式使用了该模式。例如，用户 A 在模式 C 中发出 CREATE TABLE 语句，如下所示:

```
CREATE TABLE C.X (COL1 INT)
```

当不使用两部分对象名的高位部分时，即是隐式使用该模式。当发生这种情况时，CURRENT SCHEMA 专用寄存器用于标识完成对象名的高位部分所用的模式名。CURRENT SCHEMA 的初始值是当前会话用户的授权标识。若希望在当前会话期间更改它，可使用 SET SCHEMA 语句将该专用寄存器设置为另一个模式名。

当创建数据库时，会在特定的模式内创建一些对象并将其存储在系统目录表中。

您不必显式指定要在哪个模式中创建对象；如果未指定，那么将使用语句的授权标识。例如，对于以下 CREATE TABLE 语句，模式名缺省为当前登录的授权标识（即，CURRENT SCHEMA 专用寄存器的值）：

```
CREATE TABLE X (COL1 INT)
```

动态 SQL 和 XQuery 语句通常使用 CURRENT SCHEMA 专用寄存器值来隐式限定任何非限定对象名引用。

在创建自己的对象之前，需要考虑是按自己的模式创建还是通过使用将对象按逻辑分组的另一种模式来创建。如果正在创建将共享的对象，那么使用不同的模式名将会非常有用。

## 模式名限制和建议

在命名模式时，您需要了解一些限制和建议。

- 用户定义的类型（UDT）不能具有长度超过以下位置所列示的模式长度的模式名: SQL 和 XML 限制。
- 下列模式名是保留字，不能使用: SYSCAT、SYSFUN、SYSIBM、SYSSTAT 和 SYSPROC。
- 要避免将来的潜在迁移问题，切勿使用以 SYS 开头的模式名。数据库管理器不允许您使用以 SYS 开头的模式名来创建触发器、用户定义的类型或用户定义的函数。
- 建议不要将 SESSION 用作模式名。已声明临时表必须用 SESSION 来限定。因此，可以让应用程序使用与持久表完全相同的名称来声明临时表，在这种情况下，应用程序逻辑可能会变得过分复杂。除非在处理已声明临时表，否则应避免使用模式 SESSION。

# 创建模式

在创建对象时，可以使用模式将这些对象进行分组。一个对象只能属于一种模式。使用 CREATE SCHEMA 语句来创建模式。有关模式的信息保存在连接的数据库的系统目录表中。

要创建模式并让另一个用户成为该模式的所有者（后一个操作是可选的），您需要 SYSADM 或 DBADM 权限。如果您不具有这两种权限中的任一种，那么仍可以使用您自己的授权标识来创建模式。作为 CREATE SCHEMA 语句的一部分创建的任何对象的定义者是模式所有者。此所有者可以授予和撤销其他用户的模式特权。

要通过命令行来创建模式，请输入以下语句:

```
CREATE SCHEMA <schema-name> [ AUTHORIZATION <schema-owner-name> ]
```

其中 <schema-name> 是模式的名称。此名称在目录中已记录的模式内必须唯一，并且不能以 SYS 开头。如果指定了可选 AUTHORIZATION 子句，那么 <schema-owner-name> 将成为模式所有者。如果未指定此子句，那么发出此命令的授权标识将成为模式所有者。

有关更多信息，请参阅 CREATE SCHEMA 语句。另请参阅『模式名限制和建议』。

# 复制模式

db2move 实用程序和 ADMIN_COPY_SCHEMA 过程允许您快速生成数据库模式的副本。在建立模型模式后，可以将它用作创建新版本的模板。

使用 ADMIN_COPY_SCHEMA 过程在相同数据库内复制单个模式，或将 db2move 实用程序与 **-co COPY** 操作配合使用以将单个模式或多个模式从源数据库复制至目标数据库。源模式中的大多数数据库对象被复制至新模式下的目标数据库。

## 故障诊断提示

ADMIN_COPY_SCHEMA 过程和 db2move 实用程序都调用 LOAD 命令。在处理装入时，将使数据库目标对象所在的表空间处于“备份暂挂”状态。

**ADMIN_COPY_SCHEMA 过程**

通过使用指定了 COPYNO 选项的此过程，使目标对象所在的表空间处于“备份暂挂”状态，如上面的说明中所述。要使表空间脱离“设置完整性暂挂”状态，此过程可发出 SET INTEGRITY 语句。在目标表对象定义了引用约束的情况下，也会使目标表处于“设置完整性暂挂”状态。因为该表已经处于“备份暂挂”状态，所以 ADMIN_COPY_SCHEMA 过程尝试发出 SET INTEGRITY 语句的操作将失败。

要解决这种情况，发出 BACKUP DATABASE 命令以使受影响的表空间脱离“备份暂挂”状态。接着，查看此过程生成的错误表的 `Statement_text` 列，以查找处于“设置完整性暂挂”状态的表的列表。然后，对列示的每个表都发出 SET INTEGRITY 语句，以使每个表都脱离“设置完整性暂挂”状态。

**db2move 实用程序**

此实用程序尝试复制下列类型以外的所有允许的模式对象:

- 表层次结构
- 登台表（多分区数据库环境中的 LOAD 实用程序不支持该表）
- JAR（Java™ 例程归档）
- 昵称
- 程序包
- 视图层次结构
- 对象特权（所有新对象是使用缺省权限创建的）
- 统计信息（新对象不包含统计信息）
- 索引扩展（与用户定义的结构化类型相关）
- 用户定义的结构化类型及其变换函数
- （关于模式对象的）注释

**不受支持的类型错误**

如果在源模式中检测到一个对象的类型是不受支持的类型之一，会将一个条目记录到错误文件中，表明检测到不受支持的对象类型。COPY 操作仍将成功 - 记录此条目是为了通知您此操作未复制这些对象。

**未与模式组合的对象**

在复制模式操作期间，不会对未与模式组合的对象（例如，表空间和事件监视器）进行操作。应该先在目标数据库上创建这些对象，然后再调用复制模式操作。

**已复制的表**

复制已复制的表时，不会对复制启用该表的新副本。会将该表作为常规表重新创建。

**不同的实例**

如果源数据库与目标数据库不在同一实例中，那么必须对源数据库进行编目。

**SCHEMA_MAP 选项**

使用 SCHEMA_MAP 选项来指定目标数据库上的其他模式名称时，复制模式操作将仅对对象定义语句执行最小程度的解析，以便将原始模式名称替换为新模式名称。例如，在 SQL 过程的内容中出现的原始模式的任何实例均不会替换为新模式名称。因此，复制模式操作可能无法重新创建这些对象。其他示例可能包括登台表、结果表和具体化查询表。完成复制操作后，可使用错误文件中的 DDL 来手动重新创建这些失败的对象。

**对象之间的相互依赖性**

复制模式操作尝试以满足这些对象之间的相互依赖性的顺序来重新创建对象。例如，如果表 T1 包含的某一列引用了用户定义的函数 U1，那么将先重新创建 U1，然后再重新创建 T1。但是，目录中并不随时提供过程的依赖性信息，因此重新创建过程时，复制模式操作将先尝试重新创建所有过程，然后再次尝试重新创建失败的过程（假定失败的过程依赖于上一次尝试期间成功创建的过程，于是在后续尝试期间，将成功地重新创建那些失败的过程）。在后续尝试期间，只要该操作能够成功地重新创建一个或多个失败的过程，就将继续尝试重新创建这些失败的过程。每次尝试重新创建过程时，将在错误文件中记录一个错误（和 DDL）。在错误文件中可能会看到相同过程的多个条目，即使在后续尝试期间，这些过程可能已成功创建。在完成复制模式操作后，您应该查询 SYSCAT.PROCEDURES 表以确定错误文件中列示的这些过程是否已成功地重新创建。

有关更多信息，请参阅 ADMIN_COPY_SCHEMA 过程和 db2move 实用程序。

## 使用 ADMIN_COPY_SCHEMA 过程的模式复制的示例

按下面所显示的使用 ADMIN_COPY SCHEMA 过程在同一个数据库中复制单个模式。

```
DB2 "SELECT SUBSTR(OBJECT_SCHEMA,1, 8)
  AS OBJECT_SCHEMA, SUBSTR(OBJECT_NAME,1, 15)
  AS OBJECT_NAME, SQLCODE, SQLSTATE, ERROR_TIMESTAMP, SUBSTR(DIAGTEXT,1, 80)
  AS DIAGTEXT, SUBSTR(STATEMENT,1, 80)
  AS STATEMENT FROM COPYERRSCH.COPYERRTAB"

CALL SYSPROC.ADMIN_COPY_SCHEMA('SOURCE_SCHEMA', 'TARGET_SCHEMA',
  'COPY', NULL, 'SOURCETS1 , SOURCETS2', 'TARGETTS1, TARGETTS2,
  SYS_ANY', 'ERRORSCHEMA', 'ERRORNAME')
```

此 SELECT 语句的输出如下所示:

```
OBJECT_SCHEMA OBJECT_NAME     SQLCODE     SQLSTATE ERROR_TIMESTAMP
------------- --------------- ----------- -------- --------------------------
SALES         EXPLAIN_STREAM         -290 55039    2006-03-18-03.22.34.810346

DIAGTEXT
--------------------------------------------------------------------------------
[IBM][CLI Driver][DB2/LINUXX8664] SQL0290N  Table space access is not allowed.
```

```
STATEMENT
------------------------------------------------------------------------------
set integrity for "SALES  "."ADVISE_INDEX" , "SALES"."ADVISE_MQT" , "SALES"."
1 record(s) selected.
```

## 使用 db2move 实用程序的模式复制的示例

将 db2move 实用程序与 -co COPY 操作配合使用，以便将源数据库中的一个或多个模式复制到目标数据库。在建立模型模式后，可以将它用作创建新版本的模板。

### 示例 1: 使用 -c COPY 选项

以下是 db2move -co COPY 选项的一个示例，它将模式 BAR 从样本数据库复制到目标数据库并将它重命名为 FOO:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" -u userid -p password
```

创建的新（目标）模式对象与源模式中的对象的对象名相同，但前者具有目标模式限定符。可以创建带有或不带有源表中的数据的表副本。源数据库和目标数据库可以在不同系统上。

### 示例 2: 在 COPY 操作期间指定表空间名称映射

以下示例说明如何在 db2move COPY 操作期间指定要使用的特定表空间名称映射，而不是源系统中的表空间。可指定 SYS_ANY 关键字来指示应使用缺省表空间选择算法来选择目标表空间。在本示例中，db2move 实用程序选择要用作目标的任何可用表空间:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" tablespace_map "(SYS_ANY)" -u userid -p password
```

SYS_ANY 关键字可用于所有表空间，或者可以对一些表空间指定特定映射，并对其余表空间指定缺省表空间选择算法:

```
db2move sample COPY -sn BAR -co target_db  target schema_map "
  ((BAR,FOO))" tablespace_map "((TS1, TS2),(TS3, TS4), SYS_ANY)"
  -u userid -p password
```

这指示表空间 TS1 映射至 TS2 以及 TS3 映射至 TS4，但其余表空间使用缺省表空间选择算法。

### 示例 3: 在 COPY 操作后更改对象所有者

在成功执行 COPY 操作后，可以更改在目标模式中创建的每个新对象的所有者。目标对象的缺省所有者是连接用户。如果指定了此选项，那么所有权将转移给新的所有者，如下所示:

```
db2move sample COPY -sn BAR -co target_db  target schema_map
  "((BAR,FOO))" tablespace_map "(SYS_ANY)" owner jrichards
  -u userid -p password
```

目标对象的新所有者是 jrichards。

如果源模式和目标模式位于不同的系统上，那么必须在目标系统上调用 db2move 实用程序。要将模式从一个数据库复制至另一个数据库，此操作需要将从源数据库复制的模式名称列表（用逗号分隔）和目标数据库名。

要复制模式，请从 OS 命令提示符中发出 db2move，如下所示:

```
db2move <dbname> COPY -co <COPY- options>
 -u <userid> -p <password>
```

## 重新启动失败的复制模式操作

根据正在复制的对象类型或 COPY 操作失败的阶段（即重新创建对象阶段或装入数据阶段），可以采用各种方法来处理 db2move COPY 操作期间发生的错误。

db2move 实用程序使用消息和错误文件向用户报告错误和消息。复制模式操作使用 COPYSCHEMA_<timestamp>.MSG 消息文件和 COPYSCHEMA_<timestamp>.err 错误文件。这些文件是在当前工作目录中创建的。将当前时间追加至文件名，以确保文件的唯一性。当不再需要这些消息和错误文件时，是否删除它们则取决于用户。

**注**: 可以让多个 db2move 实例同时运行。COPY 选项不返回任何 SQLCODE。这与 db2move 行为一致。

### 对象类型

可以将要复制的对象分为以下两类: 物理对象和业务对象。

物理对象指的是以物理方式存在于容器中的对象，例如，表、索引和用户定义的结构化类型。业务对象是指并不真正存在于容器中的已编目对象，例如，视图、用户定义的结构化类型（UDT）和别名。

重新创建物理对象期间发生的错误将导致实用程序回滚，而重新创建逻辑对象期间发生的错误不会这样。

### 重新启动复制模式操作

解决导致装入操作失败的问题（已在错误文件中描述）后，可使用 -tf 选项重新发出 db2move -COPY 命令，按以下语法中所示指定要复制并使用数据填充的表（在 LOADTABLE.err 文件名中传递）:

```
db2move sourcedb COPY -tf LOADTABLE.err -co TARGET_DB mytarget_db
  -mode load_only
```

还可以使用 -tn 选项手动输入表名，如以下语法所示:

```
db2move sourcedb COPY -tn "FOO"."TABLE1","FOO 1"."TAB 444",
  -co TARGET_DB mytarget_db -mode load_only
```

**注**: load_only 方式要求至少使用 -tn 或 -tf 选项来输入一个表。

## 示例

根据正在复制的对象类型或 COPY 操作失败的阶段，可以采用各种方法来处理 db2move COPY 模式操作期间发生的错误。

db2move 实用程序在下列消息和错误文件中报告模式复制错误和消息:

- COPYSCHEMA <timestamp>.MSG 消息文件
- COPYSCHEMA_<timestamp>.err 错误文件

这些文件是在当前工作目录中创建的。将当前时间追加至文件名，以确保文件的唯一性。当不再需要这些消息和错误文件时，应将它们删除。

**注**: 可以让多个 db2move 实例同时运行。COPY 选项不返回任何 SQLCODE。这与 db2move 行为一致。

**示例 1: 与物理对象相关的模式复制错误**

在目标数据库上重新创建物理对象时出现的故障都记录在错误文件 COPYSCHEMA_<timestamp>.err 中。对于每个失败对象，该错误文件包含以下信息：对象名、对象类型、DDL 文本、时间戳记和字符串格式化的 sqlca（sqlca 字段名，后跟它们的数据值）。

COPYSCHEMA_<timestamp>.err 错误文件的样本输出:

```
    1. 模式:  FOO.T1
  类型:       TABLE
错误消息:  SQL0104N 异常标记"FOO.T1"...
时间戳记:  2005-05-18-14.08.35.65
DDL:        创建视图 FOO.v1

2.模式:     FOO.T3
类型:       TABLE
错误消息:  SQL0204N FOO.V1 是未定义的名称。 时间戳记: 2005-05-18-14.08.35.68
DDL:        创建表 FOO.T3
```

如果在重新创建阶段结束时在尝试装入阶段之前记录了创建物理对象时发生的任何错误，那么 db2move 实用程序将失败并返回一个错误。目标数据库上的所有对象创建将回滚，并且会清除源数据库上内部创建的所有表。回滚将在尝试重新创建每个对象之后在重新创建阶段结束时进行，而不是在第一个故障之后进行，以便可以将所有可能的错误都收集到错误文件中。这样，在重新启动 db2move 操作之前，您将有机会修正任何问题。如果没有出现故障，将删除错误文件。

**示例 2: 与业务对象相关的模式复制错误**

在目标数据库上重新创建业务对象时出现的故障不会导致 db2move 实用程序失败。而是会将这些故障记录在 COPYSCHEMA_<timestamp>.err 错误文件中。当 db2move 实用程序完成时，可以检查故障、解决任何问题并手动重新创建每个失败的对象（为了方便起见，在错误文件中提供了 DDL）。

如果在 db2move 尝试使用 LOAD 实用程序重新填充表数据时发生错误，那么 db2move 实用程序将不会失败。而是会将一般故障信息（对象名、对象类型、DDL 文本、时间戳记和 sqlca 等等）记录到 COPYSCHEMA_<timestamp>.err 文件中，并将表的标准名称记录到另一个文件 LOADTABLE_<timestamp>.err 中。为了符合 db2move -tf 选项的格式，每个表列示在一行中，类似如下所示:

```
 "FOO"."TABLE1"
TABLE1"
 "FOO 1"."TAB 444"
```

**示例 3: 其他类型的 db2move 故障**

一些内部操作（例如，内存错误或文件系统错误）可能导致 db2move 实用程序失败。

如果在 ddl 重新创建阶段出现内部操作故障，那么将从目标模式回滚所有成功创建的对象，并且清除源数据库上所有内部创建的表（例如，DMT 表和 db2look 表）。

如果在装入阶段出现内部操作故障，那么成功创建的所有对象保留在目标模式上。装入操作期间所有遇到故障及尚未装入的表都记录在 LOADTABLE.err 错误文件中。然后，可以发出使用 LOADTABLE.err 的 db2move COPY 命令，如示例 2 中所述。如果 db2move 实用程序异常结束（例如，系统崩溃、实用程序

陷阱和实用程序停止等等），那么关于仍需要装入哪些表的信息将丢失。在这种情况下，可以使用 ADMIN_DROP_SCHEMA 过程删除目标模式并重新发出 db2move COPY 命令。

无论在尝试复制模式操作期间可能遇到什么错误，您始终可以选择使用 ADMIN_DROP_SCHEMA 过程来删除目标模式并重新发出 db2move COPY 命令。

## 删除模式

在删除模式之前，必须删除该模式中的所有对象或将它们移至另一个模式。当尝试 DROP 语句时，该模式名必须在目录中；否则会返回错误。

要使用命令行来删除模式，请输入:

```
DROP SCHEMA <name> RESTRICT
```

在以下示例中，删除了模式“joeschma”:

```
DROP SCHEMA joeschma RESTRICT
```

RESTRICT 关键字强制执行一个规则，即不能在指定的模式中为要从数据库中删除的模式定义对象，并且必须指定该关键字。

# 第 3 部分 数据库对象

逻辑数据库设计包括定义数据库对象。

可以在 DB2 数据库中创建下列数据库对象:

- 表
- 约束
- 索引
- 触发器
- 序列
- 视图

可以使用图形用户界面或通过显式执行语句来创建这些数据库对象。用于创建这些数据库对象的语句称为“数据定义语言”（DDL），它们通常以关键字 CREATE 或 ALTER 作为前缀。

了解每个数据库对象提供的特征和功能很重要，这有助于设计一个好的数据库，该数据库不仅要满足当前业务的数据存储需求，同时还应保持足够的灵活性，以便能随着时间的推移进行扩充和增长。

© Copyright IBM Corp. 1993, 2010

# 第 11 章 表

表是数据库管理器维护的逻辑结构。表由列和行组成。

每个列和行的交点处是称为*值*的特定数据项。*列*是具有相同类型的一组值，或者是具有相同类型的子类型的一组值。*行*是按一定规则排列的一系列值，以便第 *n* 个值是表中第 *n* 列的值。

应用程序可以确定将行填充到表中的顺序，但是各行的实际顺序由数据库管理器确定，通常无法控制此顺序。多维集群（MDC）有一定集群意义，但不是行之间的实际排序。

## 表的类型

根据您的环境，将需要在 DB2 数据库中创建一个或多个表来存储数据。在创建表时，需要指定表中每列的内容类型，并定义其他特征，如主键和用于强制执行业务规则的检查约束。在创建表时，还需要考虑最适合您的需求的类型。

可以使用 CREATE TABLE 语句来创建下列所有类型的表（已声明的全局临时表除外）：

**基本表** 这些类型的表将保存持久数据。

**常规表** 这些类型的表是作为堆来实现。具有索引的常规表是“常规用途”表选项。

**追加方式表**
这些类型的表是经过优化之后主要用于 INSERT 的常规表。通过使用 ALTER TABLE 语句使常规表处于追加方式。追加方式最好用于满足下列条件的表: 对于任何特定索引的集群不重要，插入速度较高，并且对表执行的删除操作不是太多甚至不执行删除操作。

**结果表** 这些类型的表由数据库管理器从一个或多个表中选择或生成的行集组成，这些行符合查询。

**摘要表** 这些类型的表是由查询定义的，也用于确定表中的数据。摘要表可用来提高查询性能。如果数据库管理器确定一部分查询可使用摘要表来解决，那么它可以重写该查询以使用摘要表。此决定基于数据库配置设置，例如，CURRENT REFRESH AGE 和 CURRENT QUERY OPTIMIZATION 专用寄存器。

**类型表** 一种表，它可以单独定义每列的数据类型，或者使类型基于用户定义的结构化类型的属性。这种表称为*类型表*。用户定义的结构化类型可以是类型层次结构的一部分。*子类型*继承其*超类型*的属性。同样，类型表可以是表层次结构的一部分。*子表*继承其*超表*的列。请注意，术语*子类型*适用于用户定义的结构化类型以及类型层次结构中这些类型下的所有用户定义的结构化类型。结构化类型 T 的*正确子类型*是类型层次结构中 T 下面的结构化类型。同样，术语*子表*适用于一个类型表以及表层次结构中该表下的所有类型表。表 T 的*正确子表*是表层次结构中 T 下面的表。

© Copyright IBM Corp. 1993, 2010

**（已声明）全局临时表**

这些类型的表（也称为用户定义的临时表）由处理数据库中的数据的应用程序使用。对数据进行处理产生的结果需要临时存储在表中。在创建全局临时表之前必须存在用户临时表空间。

**注:** 全局临时表的描述并不出现在系统目录中，因此使其对于其他应用程序而言不是持久的，也不能与其他应用程序共享此表。当使用此表的应用程序终止或与数据库断开连接时，此表中的数据被删除，此表被隐式删除。

全局临时表不支持:

- LOB 类型列（或基于 LOB 的单值类型列）
- 用户定义的类型列
- LONG VARCHAR 列
- XML 列

这些类型的表是使用 DECLARE GLOBAL TEMPORARY TABLE 语句创建的，用于代表单个应用程序保存临时数据。在应用程序与数据库断开连接时，此表将被隐式删除。

**多维集群（MDC）表**

这些类型的表是作为同时在多个键或维上进行物理集群的表来实现的。MDC 表用于数据仓储和大型数据库环境。常规表的集群索引支持数据的单维集群。MDC 表具有可以使数据集群在多个维中的优点。

**注:** MDC 是一种表类型，但是它可与分区表共存，它也可以是分区表。因此，它与其他类型的表不是互斥的。分区表还可以是 APPEND 表。不允许其他某些表类型组合，例如，MDC 与 APPEND，RCT 与任何其他表，或者其他组合。另外，MDC 在组合维中提供*有保证的集群*，而对于具有集群索引的常规表，由数据库管理器尝试进行集群，但是没有保证并且经过一段时间之后通常会降级。

**范围集群表（RCT）**

这些类型的表是作为提供快速直接访问的数据的顺序集群来实现的。表中的每一条记录都预先确定了记录标识（RID），该标识是用来在表中查找记录的内部标识。RCT 表用于数据紧密集群在一个表中的一列或多列中的情况。这些列中的最大值和最小值定义了可能值的范围。您使用这些列来访问表中的记录；这是利用 RCT 表的预确定的记录标识（RID）的最佳方法。

**分区表** 这些类型的表是使用一些数据来实现的，这些数据根据表的表分区键列中的值分布到多个数据分区中。与常规表相比，分区表简化了表数据转入和转出以及管理工作，并且提高了索引布置灵活性和查询处理效率。

对于用来保存数据的每个表，应考虑最可能满足您需要的表类型。例如: 如果具有松散集群（而不是单调增大）的数据记录，那么应考虑使用常规表和索引。如果有一些数据记录将在键中具有重复（不是唯一的）值，那么不应使用范围集群表。如果不能在磁盘上为您想要的范围集群表预分配固定的存储量，那么不应使用这种类型的表。这些因素可帮助您确定是否具有可用作范围集群表的数据。

# 设计表

在设计表时，您需要熟悉某些概念、确定表和用户数据的空间需求并确定是否将利用某些功能，例如，空间压缩和乐观锁定。

设计分区表时，您需要熟悉分区概念，例如:

- 数据组织方案
- 表分区键
- 用于在数据分区之间分配数据的键
- 用于 MDC 维的键

对于这些分区概念和其他分区概念，请参阅第 230 页的『表分区和数据组织方案』。

## 表设计概念

在设计表时，您需要熟悉一些相关概念。

### 指定列数据类型

定义列时，需要对列进行命名、定义这些列中将包含的数据的类型（称为*数据类型*）并定义要创建的表中每列的数据长度。

#### 字符数据以二进制数据形式存储

**小整数** 此数据类型用于存储精度为 15 位的二进制整数值。小整数值的范围为 -32 768 到 +32 767。小整数数据类型使用可能的最小存储空间量来存储数值（存储每个值需要 2 个字节的空间）。术语 SMALLINT 用于在表定义中声明小整数列。

**整数** 此数据类型用于存储精度为 31 位的二进制整数值。虽然整数数据类型需要的存储空间是小整数数据类型的两倍（存储每个值需要 4 个字节的空间），但它的值范围要大很多。整数值的范围为 -2 147 483 648 到 +2 147 483 647。术语 INTEGER 和 INT 可用于在表定义中声明整数列。

**大整数** 此数据类型用于在支持 64 位整数的平台上存储精度为 63 位的二进制整数值。处理作为大整数存储的较大数字比处理作为十进制值存储的类似数字效率要高很多。此外，使用大整数值执行的计算比使用实型值或双精度值执行的计算更精确。

此数据类型需要的存储空间是小整数数据类型的四倍（存储每个值需要 8 个字节的空间）。大整数的范围为 -9 223 372 036 854 775 808 到 +9 223 372 036 854 775 807。术语 BIGINT 用于在表定义中声明大整数列。

**十进制** 此数据类型用于存储同时包含整数部分和小数部分的数字；各部分组合在一起并采用压缩十进制格式存储。每次声明十进制数据类型时，必须指定精度（总位数）和小数位（数字的小数部分的位数）。十进制的精度范围为 1 到 31。可以使用以下等式计算存储十进制值所需的存储空间量: *精度*/2（已截断）+ 1 = *需要的空间字节数*。

例如，DECIMAL(8,2) 值将需要 5 个字节的存储空间（8/2 = 4; 4 + 1 = 5），而 DECIMAL(7,2) 值需要 4 个字节的存储空间（7/2 = 3.5（截断为 3）；3 + 1 = 4）。

术语 DECIMAL、DEC、NUMERIC 和 NUM 可用于在表定义中声明十进制列。

**注**: 如果没有为十进制列定义提供精度和小数位值，那么缺省情况下，使用精度值 5 和小数位值 0（因此，需要 3 个字节的存储空间）。

**单精度浮点**

此数据类型用于 32 位实数近似值。虽然单精度浮点数据类型和整数数据类型需要的存储空间量相同（存储每个值需要 4 个字节的空间），但单精度浮点数的范围要大很多: $10E^{-38}$ 到 $10E^{+38}$。

术语 REAL 和 FLOAT 可用于在表定义中声明单精度浮点列。但是，如果使用术语 FLOAT，那么指定的列长度必须在 1 与 24 之间 - FLOAT 可用来同时表示单精度和双精度浮点数据类型；指定的长度确定使用的实际数据类型。

**双精度浮点**

双精度浮点数据类型用于存储 64 位实数近似值。虽然双精度浮点数据类型需要的存储空间量与大整数数据类型相同（存储每个值需要 8 个字节的空间），但双精度浮点数的范围是可能的最大值: $-1.79760^{+308}$ 到 $-2.225E^{-307}$、0 以及 $2.225E^{-307}$ 到 $-1.79769E^{+308}$。

**固定长度字符串**

此数据类型用于存储具有不超过 254 个字符的特定长度的字符和字符串数据。术语 CHARACTER 和 CHAR 可用于声明表定义中的固定长度字符串列；每次声明固定长度字符串数据类型时，必须指定要存储的字符串数据的长度。可以使用以下等式确定存储固定长度字符串值所需的存储空间量: *固定长度* x 1 = *需要的空间字节数*。例如，CHAR(8) 值需要 8 个字节的存储空间。

**注**: 使用固定长度字符串数据类型时，如果数据的实际长度比定义列时指定的长度要小很多，那么可能会浪费存储空间。例如，如果要将值 YES 和 NO 存储在定义为 CHAR(20) 的列中。因此，为固定长度字符串列指定的固定长度应尽可能接近将存储在列中的数据的实际长度。

**可变长度字符数据**

此数据类型用于存储长度变化的字符串数据。变长字符串数据的长度最大可为 32672 个字符；但是，允许的实际长度由一个限制控制: 数据必须适合放在单个表空间页面上。这意味着对于使用 4K 页的表空间中的表，变长字符串数据的长度不能超过 4092 个字符；对于使用 8K 页的表空间中的表，变长字符串数据的长度不能超过 8188 个字符，依此类推，页大小最多为 32K。因为缺省情况下表空间是使用 4K 页创建的，所以如果要使用变长字符串数据类型来存储包含多于 4092 个字符的字符串，那么必须使用更大的页大小显式创建表空间。

**注**:

- 表行中还必须有足够的空间来容纳字符串数据。也就是说，字符串数据的存储器需求必须包括表中其他列的存储器需求，并且需要的存储空间总量不能超过表空间的页大小。

- 更新变长字符串数据值后，如果新值大于原始值，那么包含该值的记录将移至表中的另一页。这种记录称为指针记录。指针记录太多可能会导致性能急剧下降，这是因为处理单条数据记录必须检索多个页面。

术语 CHARACTER VARYING、CHAR VARYING 和 VARCHAR 可用于在表定义中声明变长字符串列。在定义变长字符串列时，必须在声明中指定应存储在该列中的最大字符数。以后存储在该列中的字符串数据值可以小于或等于指定的最大长度；如果它们更长，那么将不会存储它们，并且会返回错误。

可以使用以下等式确定存储变长字符串值所需的存储空间量：（*字符串长度* x 1）+ 4 = *需要的空间字节数*。因此，如果使用 VARCHAR(30) 数据类型存储了包含 30 个字符的字符串，那么该特定值将需要 34 个字节的存储空间。（使用此数据类型的所有字符串的长度必须小于或等于 30 个字符。）

**可变长度长字符数据**

变长长字符串数据类型也用于存储长度变化的字符串数据。此数据类型用于将长度小于或等于 32700 个字符的字符串数据存储在使用 4K 页的表空间中的表中。也就是说，在使用变长长字符串时，适用于变长字符串数据的页大小/字符串数据长度限制不适用。

术语 LONG VARCHAR 用于在表定义中声明变长长字符串列。可以使用以下等式确定存储变长长字符串值所需的存储空间量：（*字符串长度* x 1）+ 24 = *需要的空间字节数*。不推荐使用 LONG VARCHAR 和 LONG VARGRAPHIC 数据类型，将来的发行版中可能会将其除去。当为列选择数据类型时，请使用诸如 VARCHAR、VARGRAPHIC、CLOB 或 DBCLOB 之类的数据类型，因为这些数据类型在将来的发行版中会继续受到支持，并且被推荐用于可移植应用程序。

**注**：在表定义中声明列时，可以将 FOR BIT DATA 子句与任何字符串数据类型配合使用。如果使用此子句，那么在数据交换操作期间不会执行代码页转换，并且会将数据本身视为二进制（位）数据进行比较。

**字符大对象（CLOB）**

CLOB（字符大对象）值的长度最多为 2 千兆字节（2 147 483 647 字节）。CLOB 用于存储大 SBCS 或基于字符的混合（SBCS 和 MBCS）数据（例如，使用单字符集编写的文档），因此，CLOB 与 SBCS 或混合代码页关联。

**可变长度字符以二进制数据形式存储（大对象 - LOB 和二进制大对象 - BLOB）**

术语*大对象*和通用首字母缩略词 LOB 指的是 BLOB、CLOB 或 DBCLOB 数据类型。LOB 值应遵守适用于 LONG VARCHAR 值的那些限制，如“可变长度字符数据”部分中所述。即使 LOB 字符串的长度属性为 254 个字节或更少，这些限制也适用。此数据类型用于存储长度变化的二进制字符串数据。它经常用于存储非传统数据，如文档、图形图像、图片、音频和视频。

**注**：

- SQL 不能像处理其他数据那样处理二进制大对象数据。例如，不能对二进制大对象值进行排序。

**Unicode 数据**

支持的所有数据类型在 Unicode 数据库中也受支持。特别是，Unicode 数据库支持图形字符串数据，并用 UCS-2 编码存储。每个客户机（包括 SBCS 客户机）与 Unicode 数据库连接时，可使用 UCS-2 编码的图形字符串数据类型。

### 日期和时间数据（时间戳记）

*日期*数据类型用于存储由三部分组成的值（年、月和日），该值指定有效的日历数据。年部分的范围为 0001 到 9999；月部分的范围为 1 到 12；日部分的范围为 1 到 n（28、29、30 或 31），其中 n 取决于月部分以及年部分是否对应于闰年。在外部，日期数据类型显示为长度是 10 的固定长度字符串数据类型。但是，在内部，日期数据类型需要的存储空间要少很多 - 存储每个值需要 4 个字节的空间，这是因为日期值以压缩字符串形式存储。术语 DATE 用于在表定义中声明日期列。

时间数据类型用于存储由三部分组成的值（小时、分钟和秒），该值指定有效的时间（24 小时制）。小时部分的范围为 0 到 24；分钟部分的范围为 0 到 59；秒部分的范围也为 0 到 59。（如果小时部分设置为 24，那么分钟和秒部分必须设置为 0。）在外部，时间数据类型显示为长度是 8 的固定长度字符串数据类型。但是，与日期值一样，时间值以压缩字符串形式存储 - 在这种情况下，存储每个时间值需要 3 个字节的空间。术语 TIME 用于在表定义中声明时间列。

与日期一样，时间的表示在世界各地不同。因此，时间值格式还由与使用的数据库关联的地域代码确定。表 48 显示了可用的时间格式，以及它们的字符串表示的示例:

*表 48. 日期格式（YYYY = 年，MM = 月，DD = 日）*

| 格式名称 | 缩写 | 日期字符串格式 |
|---|---|---|
| 国际标准组织 | ISO | YYYY-MM-DD |
| IBM USA 标准 | USA | MM/DD/YYYY |
| IBM 欧洲标准 | EUR | MM/DD/YYYY |
| 日本工业标准 | JIS | YYYY-MM-DD |
| 特定于地点 | LOC | 取决于数据库的地域代码 |

### 数字数据

所有数字都包含符号和精度。如果某个数的值为零，那么认为符号为正号。精度是不包括符号在内的位数。请参阅 CREATE TABLE 语句的描述中的数据类型部分。

### 货币数据

版本 9.5 引入了 DECFLOAT，它是一种十进制浮点数据类型，在处理准确的十进制值的业务应用程序（例如，金融应用程序）中很有用。为十进制数据提供二进制近似值的二进制浮点数据类型（REAL 和 DOUBLE）不适合在这种应用程序中使用。DECFLOAT 不仅具有 DECIMAL 的准确性，还具有 FLOAT 的一些性能优点，这对于处理货币值的应用程序很有用。

### XML 数据

XML 数据类型用于定义表中存储 XML 值的列，这些列中存储的所有 XML 值必须是结构良好的 XML 文档。引入此本机 XML 数据类型能够将结构良好的 XML 文档以其本机分层格式存储在数据库中其他关系数据旁边。

## 生成列

生成列在表中定义，在这些列中，存储的值是使用表达式计算得出的，而不是通过插入或更新操作指定。

当创建已知始终要使用特定表达式或谓词的表时，可对该表添加一个或多个生成列。通过使用生成列，有机会在查询表数据时改进性能。

例如，当性能很重要时，以下两种表达式求值方式成本很高:

1. 必须在查询期间进行许多次表达式求值。
2. 计算很复杂。

为了改进查询的性能，可定义一个其他列，它将包含该表达式的结果。然后，当发出包括同一表达式的查询时，可直接使用生成列，或者，优化器的查询重写组件可用生成列替换该表达式。

当查询涉及连接两个或多个表中的数据时，添加生成列允许优化器选择可能更好的连接策略。

将使用生成列来改进查询的性能。结果是，可能在创建和填充表之后添加生成列。

### 示例

以下是在 CREATE TABLE 语句上定义生成列的一个示例:

```
CREATE TABLE t1 (c1 INT,
                 c2 DOUBLE,
                 c3 DOUBLE GENERATED ALWAYS AS (c1 + c2)
                 c4 GENERATED ALWAYS AS
                    (CASE WHEN c1 > c2 THEN 1 ELSE NULL END))
```

在创建此表之后，可以使用生成列来创建索引。例如,

```
CREATE INDEX i1 ON t1(c4)
```

查询可以利用生成列。例如,

```
SELECT COUNT(*) FROM t1 WHERE c1 > c2
```

可以编写为:

```
 SELECT COUNT(*) FROM t1 WHERE c4 IS NOT NULL
```

另一个示例:

```
SELECT c1 + c2 FROM t1 WHERE (c1 + c2) * c1 > 100
```

可以编写为:

```
SELECT c3 FROM t1 WHERE c3 * c1 > 100
```

## 自动编号和标识列

标识列为 DB2 提供一种方法，可自动为添加至表的每一行生成唯一数值。

当创建一个表时，如果需要唯一标识将添加至该表的每一行，那么可向该表添加一个标识列。要保证为添加至表的每一行提供唯一数字值，您应在标识列定义唯一索引，或将其声明为主键。

其他地方使用的标识列有订单号、职员编号、股票代码或者事故编号。标识列的值可以“始终”或“在缺省情况下”由 DB2 数据库管理器生成。

将对定义为 GENERATED ALWAYS 的标识列给予始终由 DB2 数据库管理器生成的值。不允许应用程序提供显式的值。定义成 GENERATED BY DEFAULT 的标识列使

应用程序能够显式地为标识列提供值。如果应用程序不提供值，那么 DB2 将生成一个值。因为由应用程序控制该值，所以 DB2 不能保证该值的唯一性。GENERATED BY DEFAULT 子句用于数据传播，其目的是复制现有表的内容；或者用于卸装和重新装入表。

在创建之后，便不能将表描述改变为包括标识列。

如果将行插入到指定了显式标识列值的表中，那么不会更新在内部生成的下一个值，并且可能会与该表中的现有值发生冲突。如果主键或在标识列定义的唯一索引在标识列强制执行值的唯一性，那么重复值将生成一条错误消息。

要对新表定义标识列，在 CREATE TABLE 语句中使用 AS IDENTITY 子句。

### 示例

以下是在 CREATE TABLE 语句上定义标识列的一个示例:

```
CREATE TABLE table (col1 INT,
                    col2 DOUBLE,
                    col3 INT NOT NULL GENERATED ALWAYS AS IDENTITY
                                      (START WITH 100, INCREMENT BY 5))
```

在本示例中，第三个列是标识列。还可以指定该列中用来在添加行时唯一标识每一行的值。在输入的第一行的列中具有值“100”；添加到该表中的每个后续行都具有相关联的值，这些值将依次增加五。

## 使用约束、缺省值和空设置约束列数据

数据通常必须符合特定限制或规则。这些限制可能适用于单条信息（例如，格式和序号），它们也可能适用于若干条信息。

**列数据值的可空性**

空值表示未知状态。缺省情况下，所有内置数据类型都支持空值的存在。但是，一些业务规则可能要求必须始终为某些列提供值，例如，紧急信息。对于这种情况，可以使用 NOT NULL 约束来确保决不会为给定表列指定空值。为特定列定义 NOT NULL 约束后，尝试在该列中放入空值的任何插入或更新操作将失败。

**缺省列数据值**

正如一些业务规则要求必须始终提供值一样，其他业务规则可能要求该值应该是什么，例如，职员的性别必须为 M 或 F。列缺省值约束用于确保在表中添加给定表列没有特定值的行时，始终为该列指定预定义的值。为列提供的缺省值可以是空值，与该列的数据类型兼容的约束值或数据库管理器提供的值。有关更多信息，请参阅: 第 213 页的『缺省列和数据类型定义』。

**键** 键是表或索引中可用来标识或访问特定数据行的单列或一组列。任何列都可以是键的一部分，并且同一列可以是多个键的一部分。由单列组成的键称为原子键；由多列组成的键称为组合键。除了具有原子或组合属性外，还根据使用键实施约束的方式对键进行了分类:

- 唯一键用来实施唯一约束。
- 主键用来实施实体完整性约束。（主键是一种特殊的唯一键，它不支持空值。）
- 外键用来实施引用完整性约束。（外键必须引用主键或唯一键；外键没有相应的索引。）

通常在声明表、索引或引用约束定义期间指定键。

**约束** 约束是对可在表中插入、删除或更新的值进行限制的规则。约束包括检查约束、主键约束、引用约束、唯一约束、唯一键约束、外键约束和参考约束。有关每种类型的约束的详细信息，请参阅: 第 243 页的第 12 章，『约束』或第 243 页的『约束的类型』。

***缺省列和数据类型定义:***

已经为某些列和数据类型预先定义或指定了缺省值。

例如，各种数据类型的缺省列值如下所示:

- *NULL*
- *0*: 用于小整数、整数、十进制、单精度浮点数和双精度浮点数。
- *空白*: 用于固定长度字符串和固定长度双字节字符字符串。
- *零长度字符串*: 用于变长字符串、二进制大对象、字符大对象和双字节字符大对象。
- *日期:* 这是插入行时的系统日期（从 CURRENT_DATE 专用寄存器获取）。将日期列添加至现有表时，将为现有行指定日期 0001 年 1 月 01 日。
- *时间或时间戳记*: 这是插入语句时的系统时间或系统日期/时间（从 CURRENT_TIME 专用寄存器获取）。将时间列添加至现有表时，将为现有行指定时间 00:00:00 或包含日期 0001 年 1 月 01 日和时间 00:00:00 的时间戳记。

  **注**: 所有行将获得给定语句的相同缺省时间/时间戳记值。
- *用户定义的单值数据类型*: 这是系统为用户定义的单值数据类型的基本数据类型定义的缺省值（强制类型转换为用户定义的单值数据类型）。

## 主键约束、引用完整性约束、检查约束和唯一约束

约束是对可在表中插入、删除或更新的值进行限制的规则。

**主键约束**

主键约束是与唯一约束具有相同属性的一个列或列的组合。可使用主键和外键约束来定义表之间的关系。

**引用完整性（或外键）约束**

外键约束（也称为引用约束或引用完整性约束）是关于一个或多个表中的一列或多列中的值的一种逻辑规则。例如，一组表共享关于公司的供应商的信息。供应商的名称有时可能会更改。可定义一个引用约束，声明表中的供应商的标识必须与供应商信息中的供应商标识相匹配。此约束会阻止可能导致丢失供应商信息的插入、更新或删除操作。

**检查约束**

（表）检查约束对添加至特定表的数据设置限制。

**唯一约束**

唯一约束（也称为唯一键约束）是这样一种规则，它禁止表的一列或多列中出现重复值。唯一键和主键是受支持的唯一约束。

## Unicode 表和数据注意事项

Unicode 字符编码标准是固定长度的字符编码方案，它包含了世界上几乎所有现用语言的字符。

有关 Unicode 表和数据注意事项的更多信息，请参阅:

- *《国际化指南》*中的『Unicode 字符编码』
- *《国际化指南》*中的『基于整理顺序的字符比较』
- *《国际化指南》*中的『基于地域代码的日期和时间格式』
- *《国际化指南》*中的『启用欧元的代码页的转换表文件』

有关 Unicode 的其他信息可在最新版本的 *The Unicode Standard* 一书中找到，并可从 Unicode 协会 Web 站点（www.unicode.org）中找到。

## 表的空间需求

在设计表时，您需要考虑表空间需求。

**长字段（LF）数据**

长字段（LF）数据存储在单独的表对象中，该对象的结构与其他数据类型的存储空间不同。

数据存储在一些大小为 32 KB 的区域中，这些区域被分成大小为 512 个字节的“2 的幂”倍的段。（因此，这些段可以是 512 个字节、1024 个字节和 2048 个字节，依此类推，直至 32768 个字节。）

长字段数据类型（LONG VARCHAR 或 LONG VARGRAPHIC）以使可用空间易于收回的方式存储。有关分配和可用空间的信息存储在 4KB 分配页中，它在整个对象中不经常出现。

对象中未使用的空间量取决于长字段数据的大小以及此大小是否在该数据的所有出现之处都是不变的。对于大于 255 个字节的数据条目，这个未使用的空间最大可为该长字段数据大小的 50%。

如果该字符数据的长度小于页大小，且它适合含有该数据其余部分的记录，那么应该使用 CHAR、GRAPHIC、VARCHAR 或 VARGRAPHIC 数据类型，而不要使用 LONG VARCHAR 或 LONG VARGRAPHIC。

**注:** 继续支持 LONG VARCHAR 和 LONG VARGRAPHIC 数据类型，但不推荐使用这两种类型，将来的发行版中可能会将其除去。

**大对象（LOB）数据**

大对象（LOB）数据存储在两个单独的表对象中，这两个对象的结构与其他数据类型的存储空间不同。要估计 LOB 数据所需的空间，需要考虑用来存储使用这些数据类型定义的数据的两个表对象:

- *LOB 数据对象:* 数据存储在一些大小为 64 MB 的区域中，这些区域被分成大小为 1024 字节的“2 的幂”倍的段。（因此，这些段可以是 1024 个字节、2048 个字节和 4096 个字节，依此类推，直至 64MB。）

  要减少 LOB 数据所用的磁盘空间量，可在 CREATE TABLE 和 ALTER TABLE 语句上的 LOB 选项子句上使用 COMPACT 参数。COMPACT 选项将所需的磁盘空间量减至最小，方法是将 LOB 数据分成更小的段。此过程不涉及数据压缩，只是使用最接近 1 KB 边界的最小空间量。使用 COMPACT 选项可能导致在追加 LOB 值时性能下降。

  包含在 LOB 数据对象中的可用空间量将受到更新和删除活动量以及要插入的 LOB 值的大小的影响。

- *LOB 分配对象:* 有关分配和可用空间的信息存储在与实际数据分离的 4 KB 分配页中。这些 4KB 页的数目取决于数据量，包括为大对象数据分配的未使用空间。开销的计算如下: 每 64 GB 一个 4 KB 页加上每 8 MB 一个 4KB 页。

  如果该字符数据的长度小于页大小，且它适合含有该数据其余部分的记录，那么应该使用 CHAR、GRAPHIC、VARCHAR 或 VARGRAPHIC 数据类型，而不要使用 BLOB、CLOB 或 DBCLOB。

**系统目录表**

创建数据库时，会创建系统目录表。当将数据库对象和特权添加至该数据库时，这些系统表将增大。

最初，这些系统表使用大约 3.5 MB 的磁盘空间。

为目录表分配的空间容量取决于表空间的类型和包含这些目录表的表空间的扩展数据块大小。例如，如果使用扩展数据块大小为 32 的 DMS 表空间，那么最初会分配给目录表空间 20 MB 的空间。注意: 对于含多个分区的数据库，目录表只位于发出 CREATE DATABASE 命令的数据库分区上。目录表的磁盘空间仅对于该数据库分区才是必需的。

**临时表** 某些语句需要临时表来进行处理（例如，使用一个工作文件来进行不能在内存中执行的排序）。这些临时表需要磁盘空间；所需的空间量取决于查询的大小、数目和属性以及返回的表的大小。

您的工作环境是独特的，这使得很难估计临时表的空间需求。例如，由于各种系统临时表的生命周期更长，为系统临时表空间分配的空间比实际在使用的表空间更多。使用 DB2_SMS_TRUNC_TMPTABLE_THRESH 注册表变量时可能会出现这种情况。

可以使用数据库系统监视器和表空间查询 API 来跟踪在正常操作期间所用的工作空间量。

可以使用 DB2_OPT_MAX_TEMP_SIZE 注册表变量来限制查询所使用的临时表空间大小。

## 表的页大小

表数据行组织成称为“页”的块。页可以具有 4 种大小: 4、8、16 和 32 千字节。表数据页不包含使用 LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB 或 DCLOB 数据类型定义的列的数据。但是，表数据页中的行包含这些列的描述符。

可以创建页大小为 4 KB、8 KB、16 KB 或 32 KB 的缓冲池或表空间。在特定大小的表空间中创建的所有表都将具有匹配的页大小。假设使用 32 KB 页大小，那么单个表或索引对象的最大大小可达 512 GB。

当使用 8 KB、16 KB 或 32 KB 页大小时，最多可有 1012 列。当使用 4 KB 的页大小时，最多可有 500 列。更大的记录标识（RID）将更改需要的内存量以及日志文件和系统临时表空间使用的空间量，它允许每个表对象有更多数据页，每页有更多记录。如果使用了最小行大小（即，12），那么页大小为 32KB 的页中最多可以包含的行数大约为 2335 行。

根据使用的页大小的不同，最大行数也会有所变化:

- 当页大小是 4 KB 时，行长度最大可为 4005 字节。

- 当页大小是 8 KB 时，行长度最大可为 8101 字节。
- 当页大小是 16 KB 时，行长度最大可为 16293 字节。
- 当页大小是 32 KB 时，行长度最大可为 32677 字节。

要确定表空间的页大小，必须考虑下列事项:

- 对于执行随机行读写操作的 OLTP 应用程序，通常最好使用较小的页大小，这样不需要的行浪费的缓冲池空间就会较少。
- 对于一次访问大量连续行的 DSS 应用程序，页大小大一些会比较好，这样就能减少读取特定数目的行所需的 I/O 请求数。但是，也有例外。如果行大小小于 `pagesize / maximum rows`，那么每一页上都会浪费空间。在这种情况下，更小一点的页大小可能更合适。

更大的页大小可允许您减少索引中的级别数。越大的页，支持的行越长。如果使用缺省页大小（4 KB），表最多可以有 500 列。较大的页大小（8 KB、16 KB 和 32 KB）支持 1012 列。表空间的最大大小与表空间的页大小成正比。

## 用户表数据的空间需求

缺省情况下，表数据存储在 4KB 页上。每一页（不管页大小如何）都包含 68 字节的数据库管理器开销。这将保留 4028 字节来存放用户数据（或行），虽然 4KB 页上的行的长度不能超过 4005 字节。一行不会横跨多页。当使用 4KB 页大小时，最多可有 500 列。

表数据页不包含用 LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB 或 DBCLOB 数据类型定义的列的数据。但是，一个表数据页中的行的确包含这些列的描述符。

通常，以“最先合适”顺序将行插入到常规表中。使用可用空间映射搜索文件，查找大小足以存放新行的第一个可用空间。更新一行时，除非该页上所剩的空间不足以包含它，否则将对它进行原地更新。如果所剩空间不足以包含新行，那么在原始行位置创建一个记录，以指向更新后的行在表文件中的新位置。

如果发出 ALTER TABLE APPEND ON 语句，那么将一直追加数据，且不保留关于数据页上任何可用空间的信息。

如果对表定义了集群索引，那么数据库管理器将尝试根据该集群索引的键顺序以物理方式建立数据的集群。将一行插入该表中时，数据库管理器将首先在集群索引中查找它的键值。如果找到了键值，那么数据库管理器就会尝试将记录插入到该键所指向的数据页上；如果找不到键值，那么将使用下一个更大的键值，以便将记录插入到包含具有下一个更大键值的记录的数据页上。如果该表中“目标”页上的空间不足，那么将使用可用空间映射来搜索邻近页以找到空间。随着时间的推移，当数据页上的空间被彻底用完时，记录将被放置在离该表中的“目标”页越来越远的位置。然后，表数据将被认为是非集群的，并且可以使用表重组来复原集群顺序。

如果表是多维集群（MDC）表，那么数据库管理器将保证始终根据一个或多个已定义的维或集群索引以物理方式集群记录。当 MDC 表是使用特定维数定义的时，将对每个维创建块索引，并且将创建用于将单元格（维值的唯一组合）映射至块的组合块索引。此组合块索引用来确定特定记录属于哪个单元格以及表中的哪些块或扩展数据块包含属于该单元格的记录。因此，当插入记录时，数据库管理器将搜索组合块索引以找到包含具有相同维值的记录的块列表，并且将搜索空间的范围仅限于这些块。如果

单元格尚不存在，或者如果单元格的现有块中空间不足，那么会将另一个块分配给该单元格，并且将记录插入其中。仍然在块中使用可用空间映射来快速找到块中的可用空间。

对于数据库中的每个用户表，可以通过如下计算公式来估计 4KB 页数:

```
ROUND DOWN(4028/(average row size + 10)) = records_per_page
```

然后，将结果插入:

```
(number_of_records/records_per_page) * 1.1 = number_of_pages
```

其中，平均行大小是平均列大小的总和，而因子“1.1”表示开销。

**注**: 此公式只是提供一个估计值。如果记录长度因碎片和溢出记录而改变，那么估计的准确性将降低。

也可以选择创建具有 8 KB、16 KB 或 32 KB 页大小的缓冲池或表空间。在特定大小的表空间中创建的所有表都将具有匹配的页大小。假设使用 32 KB 页大小，那么单个表或索引对象的最大大小可达 512 GB。当使用 8 KB、16 KB 或 32 KB 页大小时，最多可有 1012 列。对于 4KB 页大小，最大列数为 500。最大行宽也随页大小的不同而不同:

- 当页大小是 4KB 时，行长度最大可为 4005 字节。
- 当页大小是 8 KB 时，行长度最大可为 8101 字节。
- 当页大小是 16 KB 时，行长度最大可为 16293 字节。
- 当页大小是 32 KB 时，行长度最大可为 32677 字节。

更大的页大小有助于减小任何索引中的级别数。如果使用执行随机行读写的 OLTP（联机事务处理）应用程序，那么小一点的页大小会更好，这样，因意外的行而浪费的缓冲区空间更少。如果使用一次访问大量连续行的 DSS（决策支持系统）应用程序，那么大一点的页大小会更好，这样可以减少读取特定行数所需的 I/O 请求数。

不能将备份映像复原为另一种页大小。

不能导入超过 755 列的 IXF 数据文件。

已声明临时表只能采用它们自己的用户临时表空间类型创建。没有缺省用户临时表空间。临时表不能包含 LONG 数据。当应用程序与数据库断开连接时，这些表将被隐式删除，估计它们的空间需求时应考虑这一点。

## 表压缩

当表存储在磁盘上时，可以通过利用 DB2 产品中提供的表压缩功能来减少表所占用的空间。通过进行压缩，可以使用更少的数据库页来存储数据，从而节省磁盘存储空间。

并且，由于每页可以存储更多的逻辑数据，因此访问同样多的逻辑数据时需要读取的页数会更少。这意味着通过压缩还可以减少磁盘 I/O 数，这是因为每个 I/O 请求可以将更多数据保存到内存或者写入磁盘中。

对新表和现有表都可以使用压缩功能。要在 DB2 V9.5 表中实现数据压缩，可以使用两种方法:

- 行压缩（需要具备 DB2 存储优化功能部件的许可证才能获得此功能）
- 值压缩。

## 行压缩

行压缩（有时称为*深度压缩*）通过将各行之间重复的值模式替换为更短的符号字符串来压缩数据行。

使用行压缩的主要好处在于，可以在更少的空间中存储数据，从而显著降低存储成本。此外，因为您耗用存储器的速度更慢，所以将来花钱购买更多存储器的时间也会相应推迟。

除了降低成本以外，通过压缩还可以提高性能。通过执行较少的 I/O 操作就可以对已压缩的数据执行许多查询，这是因为每次从磁盘中读取时将读入更多数据。同样，可以在缓冲池中高速缓存更多数据，因此将提高缓冲池的命中率。（但是，行压缩也存在缺点，即，对数据进行压缩和解压缩需要耗用额外的 CPU 周期。）数据行压缩节省的存储量和对性能的影响与数据库中数据的特征、数据库的布局和调整以及应用程序工作负载相关。查询优化器的成本模型中包括解压缩成本。

最后，因为行压缩可以减小数据库大小，所以执行备份和复原操作将使用更少的空间，并且运行速度也更快。

本主题的其余部分将讨论下列内容:


- 第 45 页的『压缩的工作方式』
- 第 45 页的『对哪些数据进行压缩？』
- 第 45 页的『启用或禁用行压缩』
- 第 45 页的『UPDATE 活动对日志和已压缩表的影响』.


### 压缩的工作方式

行压缩使用基于静态字典的压缩算法来逐行压缩数据。字典用来将表行中重复的字节模式映射至更小的符号；这时，这些符号替换表行中更长的字节模式。压缩字典与表数据行一起存储在表的数据对象部分。

### 对哪些数据进行压缩?

可对存储在基本表行和日志记录中的数据进行行压缩。

**限制**: 未压缩 XML 列中的数据。

### 启用或禁用行压缩

要使用行压缩，您必须具有 DB2 存储优化功能部件的许可证。通过将表的 COMPRESS 属性设置为 YES 来压缩表数据。可以在创建表时通过在 CREATE TABLE 语句中包括 **COMPRESS YES** 选项来设置此属性；也可以在 ALTER TABLE 语句中使用此选项将现有表改变为使用压缩功能。启用压缩功能之后，用于对表添加数据的操作（例如，INSERT、LOAD INSERT 或 IMPORT INSERT 操作）可以使用行压缩功能。

**要点**: 对表启用行压缩时，就对整个表启用了行压缩，即使此表由多个表分区组成也是如此。

要对表禁用压缩功能，请使用带有 **COMPRESS NO** 选项的 ALTER TABLE 语句；将不会对您以后添加的行进行压缩。要对整个表进行解压缩，必须使用 REORG TABLE 命令来执行表重组。

## UPDATE 活动对日志和已压缩表的影响

根据 UPDATE 活动以及数据行中更新更改的位置，可能会增加日志的使用。有关更新日志记录和列顺序的信息，请参阅『第 221 页的『对列进行排序以使更新日志记录最少』』。

如果行大小增大，那么该行的新版本可能不适合放在当前数据页上。而且，行的新映像将存储在溢出页上。要将创建指针溢出记录的次数降到最低限度，应增大每一页中在使用 ALTER TABLE 语句的 **PCTFREE** 选项进行重组之后保留的可用空间所占的百分比。例如，如果在启用压缩功能之前 PCTFREE 设置为 5%，那么可以在启用压缩功能时将它更改为 10%。对于频繁更新的数据来说，此建议尤其重要。

*压缩字典:*

数据库管理器将对每个支持行压缩的表创建压缩字典。此字典用来将表行中重复的字节模式映射至更小的符号；这时，这些符号替换表行中更长的字节模式。

行压缩逻辑将对表进行扫描以查找重复的数据。将检查所有行（而不仅仅是检查这些行的某些字段或者某些部分），以了解是否存在重复的条目或模式。收集重复的条目之后，DB2 数据库将构建一个压缩字典，并为这些条目指定简短的数字键。包含文本数据的表可以具有重复的字符串、具有重复字符的数据以及前导或尾部空格。与包含数字数据的表相比，数据库管理器可以更有效地压缩这些表。

用于压缩查询和解压查询的字典存储在数据库中的隐藏对象中，并且已将它高速缓存在内存中以便快速访问。字典不会占用很多空间。即使对于非常大的表，压缩字典通常也只占用大约 100 KB 空间。

*创建压缩字典:*

可以自动构建或手动构建为行压缩而启用的表的压缩字典。

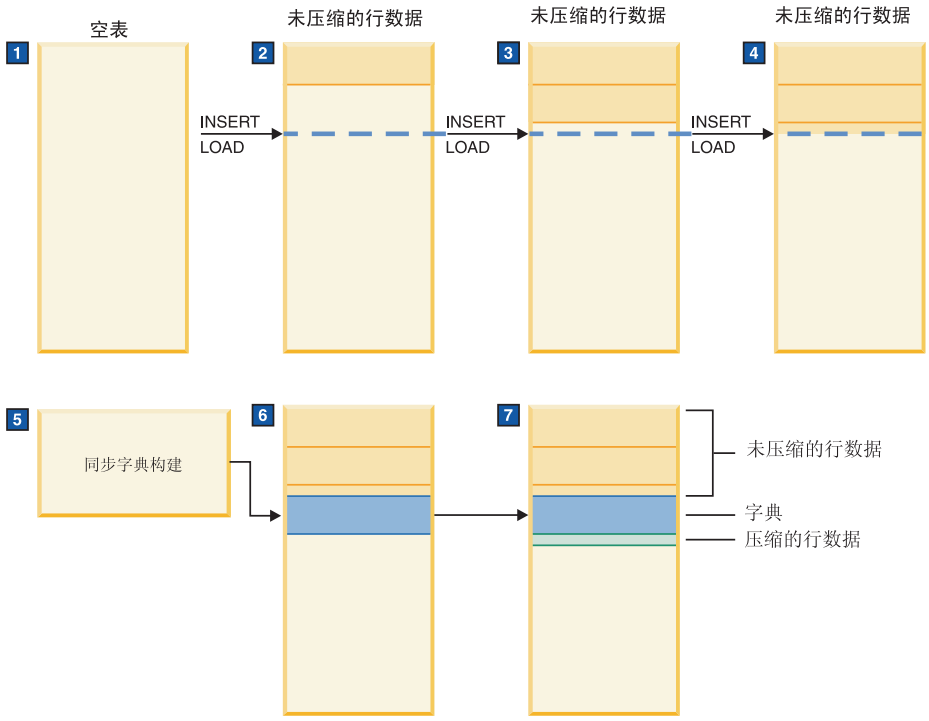
### 自动创建字典

从 DB2 版本 9.5 开始，如果满足下列每个条件，那么会自动创建压缩字典:

- 表的 COMPRESS 属性设置为 YES。当您创建表时，可以使用 CREATE TABLE 语句的 **COMPRESS YES** 选项将此属性设置为 YES；还可以在 ALTER TABLE 语句中使用该选项将现有表改变为使用压缩功能。
- 该表尚不存在压缩字典
- 表的大小达到一定程度，有足够的数据用于构造重复数据的字典。

如果仍然启用了压缩，那么就会使用该压缩字典对后来移入该表的数据进行压缩。

下图显示了自动创建压缩字典的过程:




1. 由于表是空的而未创建压缩字典。
2. 数据是使用插入或装入操作插入表中的，并且保持未压缩。
3. 随着更多数据被插入或装入表中，它保持未压缩。
4. 如果 COMPRESS 属性设置为 YES，那么在达到阈值之后就会自动触发字典的创建过程。
5. 创建了字典。
6. 已将字典追加至表。
7. 从此时起，数据就是压缩的。

**要点**：仅对随后插入该表中以创建字典的行进行压缩。对于创建该字典之前已存在的行，除非更改了这些行，或者手动重建了该字典，否则它们将保持未压缩状态。

**限制**：未压缩 XML 列中的数据。

### 手动创建字典

尽管支持压缩的表增大到足够大时会自动创建字典，但是如果不存在压缩字典，您也可以强制创建压缩字典，或者使用带有 **RESETDICTIONARY** 选项的 REORG TABLE 命令来复位现有压缩字典。如果该表中至少有一行数据，此命令就会强制创建压缩字典。对表进行重组是一项脱机操作；使用“自动创建字典”功能的一项好处是构建字典时该表仍然处于联机状态。

您可以使用带有 **ROWCOMPESTIMATE** 选项的 INSPECT 命令，而不使用 REORG TABLE 命令来强制创建新的字典。如果该表尚无压缩字典，那么此命令将创建一个新的

压缩字典。此方法的优势在于该表仍然处于联机状态。但是，会对您随后添加的行进行压缩，在运行 INSPECT 命令之前已存在的行将保持未压缩状态，直到您对表执行重组为止。

### 复位压缩字典

无论是自动还是手动创建压缩字典，字典均为静态；在构建之后也不会更改。当您添加或更新行时，将根据当前压缩字典中存在的数据来压缩这些行。此行为在许多情况下都存在。例如，以用于维护城市自来水公司客户帐户的数据库中的表为例。这样一个表中可能有 STREET_ADDRESS、CITY、PROVINCE、TELEPHONE_NUM、POSTAL_CODE 和 ACCOUNT_TYPE 等列。如果使用这种表中的数据构建了压缩字典，即使它只是一个适当大小的表，也可能存在足够多的重复信息，因此对它进行行压缩将节省大量空间。这是因为不同客户之间可能有许多共同数据，例如 CITY、POSTAL_CODE 和 PROVINCE 都相同，而 STREET_ADDRSS 或 TELEPHONE_NUM 列的某些部分相同。

但是，其他表可能会随时间推移而显著更改。以一个用于存储零售数据的范围分区表为例，如下所示:

- 每个分区将存储一年中特定月份的数据，
- 在月末，使用 ALTER TABLE 语句的 ATTACH PARTITION 子句将具有所给定月份的销售数据的分区转入该表中。

这种情况下，在某个月份（比如说，四月份）创建的压缩字典可能不会反映后续月份重复的销售数据。在表中数据随时间推移而显著更改的情况下，您可能希望使用带有 **RESETDICTIONARY** 选项的 REORG TABLE 命令来复位压缩字典。

#### *对列进行排序以使更新日志记录最少:*

使用 CREATE TABLE 语句定义列时，请考虑列的顺序，尤其是对于更新密集型工作负载。应将频繁更新的列分组在一起，并在表定义的末尾进行定义。这将能够提高性能，使得日志记录较少的字节数并写入较少的日志页，同时使执行大量更新的事务需要较少的活动日志空间。

数据库管理器不会自动假定在 UPDATE 语句的 SET 子句中指定的列的值正在更改。为了限制索引维护和需要日志记录的行数，数据库将新列值与旧列值进行比较，以确定列是否正在更改。如果正在更改列中的值，那么认为仅在更新这些列。当列数据存储在数据行（long、LOB、ADT 和 XML 列类型）外部时，或者对固定长度的列启用注册表变量 DB2ASSUMEUPDATE 时，此 UPDATE 行为会有所不同。对于这些例外情况，认为正在更改列值，因此不比较新列值和旧列值。

有三种不同类型的 UPDATE 日志记录。

- 完整行映像前后日志记录。将对行映像前后的整个过程进行日志记录。这是对启用了 DATA CAPTURE CHANGES 的表执行的唯一日志记录类型，它将导致日志记录行更新的大多数字节数。
- 完整 XOR 日志记录。此 XOR 从更改的第一个字节到较短行的末尾，然后直到较长行的任何剩余字节在行映像前后都不同。这将导致它记录的字节数比完整行映像前后日志记录所记录的字节数要少，并且数据字节数比作为最大行映像大小的日志记录头信息要多。

- 部分 XOR 日志记录。此 XOR 从更改的第一个字节到更改的最后一个字节在行映像前后都不同。字节位置可能是某列的第一个字节或最后一个字节。这将导致日志记录的字节数最少，但却是最有效的行更新的日志记录类型。

未对表启用 DATA CAPTURE CHANGES 时，日志记录的更新数据量取决于:

- 已更新的列的相似性（COLNO）
- 已更新的列是固定长度还是可变长度
- 是否启用了行压缩（COMPRESS YES）

当行的总长度未更改时，即使启用了行压缩，数据库管理器也会计算并写入最佳部分 XOR 日志记录。

当行的总长度正在更改时（在更新可变长度的列并且启用了行压缩时通常如此），数据库管理器将确定要更改的第一个字节以及写入完整 XOR 日志记录。

## 值压缩

值压缩针对数据的表示以及数据库管理系统内部用来存储数据的存储器结构来优化空间使用情况。值压缩涉及除去一个值的重复条目，仅存储一个副本。存储的副本记录对存储值的任何引用的位置。

创建表时，可使用 CREATE TABLE 语句的可选 VALUE COMPRESSION 子句来指定表将使用值压缩功能。通过使用 ALTER TABLE 语句的 ACTIVATE VALUE COMPRESSION 子句，可以在现有的表中启用值压缩功能。要在表中禁用值压缩功能，请使用 ALTER TABLE 语句的 DEACTIVATE VALUE COMPRESSION 子句。

使用 VALUE COMPRESSION 时，不会将已指定给已定义的变长数据类型（VARCHAR、VARGRAPHICS、LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB 和 DBCLOB）的 NULL 和零长度数据存储在磁盘上。只有与这些数据类型相关联的开销值才会占用磁盘空间。

如果使用了 VALUE COMPRESSION，那么还可以使用可选 COMPRESS SYSTEM DEFAULT 选项来进一步减少磁盘空间的使用量。如果插入的或更新的值等于列的数据类型的系统缺省值，那么使用的磁盘空间最少，这是因为，缺省值将不会存储在磁盘上。支持 COMPRESS SYSTEM DEFAULT 的数据类型包括所有数字类型列、定长字符和定长图形字符串数据类型。这表示零和空格可以压缩。

使用值压缩功能时，行中压缩列的字节数可能大于同一列的未压缩版本。如果行大小接近页大小所允许的最大值，那么必须确保压缩列和未压缩列的字节数之和不超出表空间中该表所允许的行长度。例如，在页大小为 4 KB 的表空间中，允许的行长度是 4005 字节。如果超出了可允许行长度，那么会返回错误消息 SQL0670N。用于确定压缩列和未压缩列的字节数的公式已记录在 CREATE TABLE 语句的文档中。

如果停用值压缩功能:

- 如果先前已启用 COMPRESS SYSTEM DEFAULTS，那么还将以隐式方式将其停用
- 未压缩的列可能会导致行大小超出当前表空间的当前页大小所允许的最大值。如果发生这种情况，那么将返回错误消息 SQL0670N。
- 现有的已压缩数据将保持处于压缩状态，直到该行被更新或者您使用 REORG 命令来执行表重组为止。

# 乐观锁定

对于版本 9.5，增强的乐观锁定支持提供了一项技术，用于在选择行与更新或删除行之间未拥有行锁定的 SQL 数据库应用程序。

编写应用程序时，可以乐观地假定在更新或删除操作前未锁定的行不可能更改。如果行更改了，那么更新或删除操作将失败，但应用程序逻辑可以处理这种故障，例如，通过重试删除操作。

此增强的乐观锁定的优点是提高了并行性，因为其他应用程序可以读写相同的行。在业务交易与数据库事务无关的三层环境中，由于不能在业务交易间保持锁定，所以使用此乐观锁定技术。

## 乐观锁定

*乐观锁定*是一项技术，它用于在选择行与更新或删除行之间未拥有行锁定的 SQL 数据库应用程序。

编写应用程序时，乐观地假定在更新或删除操作前未锁定的行不可能更改。如果行更改，那么更新或删除操作将失败，并且应用程序逻辑将通过重试选择操作（此处是举例说明）来处理这种故障。乐观锁定的一个优点是提高了并行性，因为其他应用程序可以读写该行。在业务交易与数据库事务无关的三层环境中，由于不能在业务交易间保持锁定，所以使用乐观锁定技术。

但是，按值乐观锁定具有一些缺点:

- 在没有其他数据服务器支持的情况下可能导致*主动错误信息*，它是使用乐观锁定时出现的一种情况，表示在选择后*已更改*的行必须重新选择才能进行更新。（这与*被动错误信息*不同，它表示在选择后*未更改*的行必须重新选择才能进行更新。）
- 在应用程序中需要更多的重试逻辑
- 构建 UPDATE 搜索条件对应用程序来说很复杂
- DB2 服务器根据值来搜索目标行的效率不高
- 某些客户机类型与数据库类型之间的数据类型不匹配（例如，时间戳记）不允许在搜索式 UPDATE 中使用所有列

版本 9.5 增加了对更容易且速度更快的乐观锁定的支持，这种乐观锁定不会产生*主动错误信息*。此支持是通过使用下列新的 SQL 函数、表达式和功能增加的:

- 行标识（RID_BIT 或 RID）内置函数
- ROW CHANGE TOKEN 表达式
- 基于时间的更新检测
- 隐式隐藏的列

DB2 应用程序可通过构建搜索式 UPDATE 语句启用*按值乐观锁定*，该语句查找具有与所选值完全相同的值的行。如果行的列值已更改，那么搜索式 UPDATE 语句将失败。

使用此编程模型的应用程序将从增强的乐观锁定功能中获得好处。请注意，未使用此编程模型的应用程序不会被视为乐观锁定应用程序，它们将继续和以前一样工作。

### 行标识（RID_BIT 或 RID）内置函数

可以在选择列表或谓词语句中使用此内置函数。在谓词中，例如，WHERE RID_BIT(tab)=?，RID_BIT 等号谓词是作为一种新的直接访问方法来实现的，以

便有效地找到行。以前，这种值“使用值的乐观锁定”是通过将所有选择的列值添加至谓词并依赖于某些独特列组合来仅限定单个行完成的，这是一种效率较低的访问方法。

**ROW CHANGE TOKEN 表达式**

此新表达式将返回 BIGINT 形式的标记。标记表示行的修改序列中的一个相对点。应用程序可以将行的当前行更改标记值与上次访存行时存储的行更改标记值进行比较，以确定该行是否已更改。

**基于时间的更新检测:**

此功能是使用 ROW CHANGE TIMESTAMP 表达式添加至 SQL 的。要支持此功能，需要在表中定义一个新生成的行更改时间戳记列，以用于存储时间戳记值。可以使用 ALTER TABLE 语句将此列添加至现有表，或者在创建新表时定义行更改时间戳记列。行更改时间戳记列的存在也会影响乐观锁定的行为，因为该列用来将行更改标记的详细程度从页级别提高到行级别，这样可以为乐观锁定应用程序带来极大好处。DB2 z/OS® 版中也已增加此功能。

**隐式隐藏的列:**

为了兼容，此功能不需要对现有表和应用程序使用行更改时间戳记列。使用隐式列列表时，未外部化隐式隐藏的列。例如:

- 对表执行 SELECT * 不会在结果表中返回隐式隐藏的列
- 不使用列列表的 INSERT 语句不期望隐式隐藏的列的值，但应将该列定义为允许空值或具有另一个缺省值。

**注:** 请参阅 DB2 词汇表以了解乐观锁定术语（例如，*乐观并发控制*、*悲观锁定*、*ROWID* 和*更新检测*）的定义。

## 乐观锁定限制和注意事项

本主题列示您需要了解的乐观锁定限制。

- 下列键、列和名称中不支持行更改时间戳记列（如果使用此列，那么将返回 sqlstate 429BV）:
  - 主键
  - 外键
  - 多维集群（MDC）列
  - 范围分区列
  - 数据库散列分区键
  - DETERMINED BY 约束列
  - 昵称
- 数据库分区功能部件（DPF）配置中不支持 RID() 函数。
- 在乐观锁定方案中，在访存操作与更新操作之间执行的联机或脱机表重组可能导致更新操作失败，但一般应用程序重试逻辑可以处理这种情况。
- 在版本 9.5 中，IMPLICITLY HIDDEN 属性仅限于乐观锁定的 ROW CHANGE TIMESTAMP 列。
- 现场重组限于其中的 ROW CHANGE TIMESTAMP 列已添加到现有表中的表，直到保证所有行都已具体化为止（对此错误返回 SQL2219 和原因码 13）。使用 LOAD REPLACE 命令或传统表重组可以完成此任务。这将防止产生*主动错误信息*。创建时定义了 ROW CHANGE TIMESTAMP 列的表没有限制。

### 隐式隐藏的列的注意事项

在选择列表中指定 * 的查询的结果表中不包括定义为 IMPLICITLY HIDDEN 的列。但是，可以在查询中显式引用隐式隐藏的列。

如果在插入时未指定列列表，那么 VALUES 子句或插入的选择列表不应包括此列（通常，它必须是生成列、可缺省的列或可空列）。

例如，可以在选择列表中或查询谓词中引用隐式隐藏的列。此外，可以在 CREATE INDEX 语句、ALTER TABLE 语句、INSERT 语句、MERGE 语句或 UPDATE 语句中显式引用隐式隐藏的列。可以在引用约束中引用隐式隐藏的列。未包含列列表的 REFERENCES 子句隐式引用父表的主键。父表的主键可能包含定义为隐式隐藏的列。允许这种引用约束。

- 如果具体化查询定义的全查询的选择列表显式引用隐式隐藏的列，那么该列将是具体化查询表的一部分。否则，隐式隐藏的列不是具体化查询表的一部分，该具体化查询表引用包含隐式隐藏的列的表。
- 如果视图定义（CREATE VIEW 语句）的全查询的选择列表显式引用隐式隐藏的列，那么该列将是视图的一部分（但视图列不被视为“隐藏的”）。否则，隐式隐藏的列不是视图的一部分，该视图引用包含隐式隐藏的列的表。

### 基于标号的访问控制（LBAC）的注意事项

当列受 LBAC 保护时，用户是否能访问该列由 LBAC 策略和该用户的安全标号确定。如果此保护适用于行更改时间戳记列，那么它将通过派生自该列的 ROW CHANGE TIMESTAMP 和 ROW CHANGE TOKEN 表达式扩展为引用该列。

因此，在确定表的安全策略时，请确保适当时需要使用乐观锁定或基于时间的更新检测的所有用户可以访问行更改时间戳记列。请注意，如果没有行更改时间戳记列，那么 LBAC 不能阻塞 ROW CHANGE TOKEN 表达式。但是，如果改变表以添加行更改时间戳记列，那么任何 LBAC 注意事项都将适用。

## 行更改标记和*被动错误信息*的详细程度

RID_BIT() 内置函数和行更改标记是乐观锁定的唯一要求。但是，表模式也会影响乐观锁定的行为。

例如，使用下面显示的任一语句子句定义的行更改时间戳记列导致 DB2 服务器存储最后一次更改（或最初插入）行的时间。这提供了一种方法来捕获行的最新更改的时间戳记。这是时间戳记列，它由数据库管理器维护，除非使用 GENERATED BY DEFAULT 子句来接受用户提供的输入值。

```
GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
GENERATED BY DEFAULT FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP
```

因此，当应用程序对表使用新的 ROW CHANGE TOKEN 表达式时，需要考虑两种可能性:

- *表不包含行更改时间戳记列:* ROW CHANGE TOKEN 表达式返回派生的 BIGINT 值，该值由同一页面上的所有行共享。如果更新页面上的一行，那么该页面上所有行的行更改标记都会更改。这表示在对其他行进行更改时，更新操作可能会失败，这是一种称为被动错误信息的属性。

**注:** 仅当应用程序可以容忍*被动错误信息*并且不想对 ROW CHANGE TIMESTAMP 列的每行添加其他存储器时，才使用此方式。

- *表包含行更改时间戳记列:* ROW CHANGE TOKEN 表达式返回从该列中的时间戳记值派生的 BIGINT 值。在这种情况下，*被动错误信息*可能但较少出现: 如果重组或重新分发表，那么在移动行和应用程序使用先前的 RID_BIT() 值时，可能会出现*被动错误信息*。

## 基于时间的更新检测

某些应用程序需要知道特定时间范围内的数据库更新，这些更新可用于复制数据、审计方案等等。新的 ROW CHANGE TIMESTAMP 表达式提供此信息。

它返回用类似于 CURRENT TIMESTAMP 的本地时间表示的时间戳记，表示上次更新行的时间。对于已更新的行，此时间戳记反映行的最新更新时间。否则，值对应于行的原始插入时间。

对于数据库或表分区的每个表的每一行来说，ROW CHANGE TIMESTAMP 的值都是唯一的。即，并不是每个数据库分区的所有行都是唯一的，只有同一个表中的行才是唯一的。值表示行的修改顺序。更迟修改的行始终比较早修改的行具有更新的值。由于值始终会从较早的值变为更新的值，因此在下列情况下它可能会与系统不同步:

- 系统时钟已更改
- 行更改时间戳记列是 GENERATED BY DEFAULT（仅用于数据传播）并且为行提供了不同步的值

使用 ROW CHANGE TIMESTAMP 表达式的先决条件是表必须定义行更改时间戳记列。每行都返回插入或最后一次更新该行的时间戳记。通过下列两种方法可以使行更改时间戳记列成为表的一部分:

- *创建表时定义行更改时间戳记列。* ROW CHANGE TIMESTAMP 表达式返回该列的值。对于此类别，时间戳记是精确的。通常，由数据库生成的行更改时间戳记受到插入速度的限制，还有可能受到时钟处理（包括 DST 调整）的限制。
- *创建表时未定义行更改时间戳记列，*但以后通过 ALTER TABLE 语句添加了该列。ROW CHANGE TIMESTAMP 表达式返回该列的值。对于此类别，先前改变的旧行不包含实际时间戳记，直到第一次更新它们或执行脱机表重组为止。

  **注:** 时间戳记是数据库中实际进行更新的大致时间，从当时的系统时钟起并考虑时间戳记不能在数据库/表分区中重复的局限性。实际上，这通常是更新时间的非常准确的表示。通常，由数据库生成的行更改时间戳记受到插入速度的限制，还有可能受到时钟处理（包括 DST 调整）的限制。

  在执行 ALTER TABLE 语句后尚未更新的行将返回该列的类型缺省值，它为 0001 年 1 月 01 日午夜。只有已更新的行才具有唯一时间戳记。已通过脱机表重组具体化其时间戳记的行将返回在重组表期间生成的唯一时间戳记。使用 INPLACE 选项进行的重组不够，因为它不具体化模式更改。

在任一情况下，在执行重新分发时也可能会更新行的时间戳记。如果在重新分发期间将行从一个数据库分区移至另一个数据库分区，那么必须生成新的时间戳记，并保证该时间戳记在目标中是唯一的。

## 为 ROW CHANGE TIMESTAMP 生成的时间值

由于强制使每个分区具有唯一的值，所以为行更改时间戳记列生成的准确值存在一些边界条件。

每次向后调整系统时钟以在 DB2 服务器上进行时钟校正或实现夏令时策略时，显示的时间戳记相对于当前系统时钟值或 CURRENT TIMESTAMP 专用寄存器值来说可能会超前一些。如果时间戳记是在调整系统时钟之前生成的（也就是说，比调整后的时间要晚），那么将会出现这种情况，这是因为时间戳记始终按升序生成以保持唯一性。

如果列是通过 REORG 操作或在 LOAD 操作过程中添加至表的，那么为这种列生成时间戳记时，将在处理实用程序的某个时刻从初始时间戳记值开始按顺序生成时间戳记。如果实用程序处理行的速度比时间戳记粒度要快（也就是说，每秒处理的行数超过 1 百万行），那么为某些行生成的值相对于系统时钟或 CURRENT TIMESTAMP 专用寄存器来说也可能会超前一些。

在任一情况下，一旦系统时钟赶上行更改时间戳记值，就会有一个插入行的近似时间。在此时间之前，将按时间戳记类型所允许的最细粒度以升序顺序生成时间戳记。

## RID_BIT() 和 RID() 内置函数

可以对表中的每行选择 RID_BIT() 和*行更改标记*。可以在应用程序需要的任何隔离级别进行选择。

应用程序可以通过执行下列操作来使用乐观锁定更新相同（未更改的）行:

- 搜索 RID_BIT() 以直接访问（不扫描）更新目标行
- 搜索行更改标记以确保这是同一未更改的行

可以在选择后的任何时刻在同一工作单元或者甚至跨连接边界来执行此更新（或删除）；唯一的要求是在某个时间点获取给定行的上面两个值。

“面向 WebSphere 的编程模型”中使用乐观锁定。例如，Microsoft .NET 使用此模型来处理后跟 UPDATE 或 DELETE 语句的 SELECT 语句，如下所示:

- 连接至数据库服务器并从表中选择期望的行
- 断开与数据库的连接，或者释放行锁定以便其他应用程序可以读取、更新、删除和插入数据，而不会由于应用程序拥有的锁定和资源而产生任何并发冲突（“未落实的读取”隔离允许较高的并行性并假定其他应用程序落实它们的更新和删除事务，然后此乐观锁定应用程序将读取已更新的值，这样乐观搜索式 UPDATE/DELETE 将成功）
- 对选择的行数据执行某些本地计算
- 重新连接至数据库服务器，然后在一个或多个特定目标行上搜索更新或删除操作（如果目标行已更改，那么将处理失败的 UPDATE 或 DELETE 语句）

使用此编程模型的应用程序将从增强的乐观锁定功能中获得好处。请注意，未使用此编程模型的应用程序不会被视为乐观锁定应用程序，它们将继续和以前一样工作。

### RID_BIT() 和 RID() 内置函数的功能

以下是将对增强的乐观锁定和更新检测实施的新功能:

**RID_BIT( <table designator> )**

一个新的内置函数，它返回行的记录标识（RID）作为 VARCHAR(16) FOR BIT DATA。

**注**: DB2 z/OS 版实施返回类型为 BIGINT 的内置函数 RID，但该 RID 对于 Linux、UNIX 和 Windows RID 来说不够大。为了实现兼容，此 RID() 内置函数不仅返回 RID_BIT()，还返回 BIGINT。

此 RID() 内置函数不在 DPF 环境中工作，并且不包含表版本信息。否则，它与 RID_BIT 的工作方式相同。只有在编写将移植到 z/OS 服务器的应用程序时，才应使用此内置函数。除有必要，否则本主题仅仅指的是 RID_BIT。

**RID_BIT() 内置函数**

可以在选择列表或谓词语句中使用此内置函数。在谓词中，例如，`WHERE RID_BIT(tab)=?`，RID_BIT 等号谓词是作为一种新的直接访问方法来实现的，以便有效地找到行。以前，这种值*使用值的乐观锁定*是通过将所有选择的列值添加至谓词并依赖于某些独特列组合来仅限定单个行完成的，这是一种效率较低的访问方法。

**ROW CHANGE TOKEN FOR <table designator>**

一个新表达式，它返回 BIGINT 形式的标记。标记表示行的修改序列中的一个相对点。应用程序可以将行的当前行更改标记值与上次访存行时存储的行更改标记值进行比较，以确定该行是否已更改。

**ROW CHANGE TIMESTAMP 列**

缺省类型为 TIMESTAMP 的 GENERATED 列，可以将它定义为:

```
GENERATED ALWAYS FOR EACH ROW ON UPDATE
AS ROW CHANGE TIMESTAMP
```

或者（建议仅用于数据传播或卸装和重新装入操作）:

```
GENERATED BY DEFAULT FOR EACH ROW ON UPDATE
AS ROW CHANGE TIMESTAMP
```

每次更改行时，此列中的数据就会更改。定义此列时，将从该列中派生 ROW CHANGE TOKEN 值。请注意，在使用 GENERATED ALWAYS 时，数据库管理器将确保此值在数据库分区或表分区中是唯一的，以确保不可能产生*主动错误信息*。

要使用前两个元素 RID_BIT 和 ROW CHANGE TOKEN，不需要对数据库模式进行其他更改。但是，请注意，如果没有 ROW CHANGE TIMESTAMP 列，那么行更改标记将由同一页面上的每行共享。更新该页面上的任何行可能导致对存储在同一页面上的其他行产生*被动错误信息*。如果包含此列，那么 ROW CHANGE TOKEN 将从时间戳记派生，并且不由表或数据库分区中的任何其他行共享。请参阅第 225 页的『行更改标记和*被动错误信息*的详细程度』。

### 基于时间的更新检测功能

一个返回时间戳记值的新表达式，该时间戳记值表示表标志符所标识的表中的行最后一次更改的时间。

```
ROW CHANGE TIMESTAMP FOR <table designator>
```

不包含 ROW CHANGE TIMESTAMP 列的表不支持 ROW CHANGE TIMESTAMP 表达式。

ROW CHANGE TIMESTAMP 表达式仅用于基于时间的更新检测方案，并且要求为表标志符所标识的表定义行更改时间戳记列。此列由数据库管理器管理，并且用于存储

ROW CHANGE TIMESTAMP 表达式返回的时间戳记值。此时间戳记与 CURRENT TIMESTAMP 不同，因为在数据库对每个数据库分区中的每行指定此时间戳记时保证了它的唯一性。它是插入或更新的每个单独行修改时间的本地时间戳记近似值。

**注**: 即使这两个功能（即，RID_BIT() 和 RID() 内置函数以及基于时间的更新检测功能）内部相关，但一定要注意不能交换使用 ROW CHANGE TOKEN 和 ROW CHANGE TIMESTAMP 表达式；具体地说，ROW CHANGE TIMESTAMP 表达式不是乐观锁定用法中的一部分。

## 计划启用乐观锁定

由于可以在未对涉及的表进行 DDL 更改的情况下使用新的 SQL 表达式和乐观锁定的属性，所以可以很容易在测试应用程序中尝试乐观锁定。

请注意，如果未进行 DDL 更改，那么乐观锁定应用程序可能会比进行 DDL 更改时获得更多*被动错误信息*。由于被动错误信息可能会导致太多次尝试，所以获得被动错误信息的应用程序在生产环境中可能不会缩放良好。因此，要避免产生被动错误信息，乐观锁定目标表应该:

- 在创建时定义 ROW CHANGE TIMESTAMP 列，或者
- 改变后包含 ROW CHANGE TIMESTAMP 列

如果进行了建议的 DDL 更改，那么产生的被动错误信息将会少很多。唯一的被动错误信息是由于诸如重组之类的表级别操作而产生的，而不是由于对不同行操作的并发应用程序产生的。

通常，数据库管理器允许被动错误信息（例如，联机或脱机重组），并且行更改时间戳记列的存在足以确定正在使用行级别粒度还是页级别粒度。还可以查询 SYSCAT.COLUMNS 以找到一个表，该表包含 ROWCHANGETIMESTAMP 列的值为 YES 的行。

例如，如果每个页面有一行，或者如果很少或从不在同一数据页面上执行更新和删除操作，那么彻底分析应用程序和数据库后可能指示此 DDL 不是必需的。这种分析是异常。

对于更新时间戳记检测用法，您必须对表的 DDL 进行更改，并且可能需要重组表以具体化值。如果担心这些更改可能会对生产数据库产生负面影响，那么首先应在测试环境中建立更改的原型。例如，多余的列可能影响行大小局限性和计划的选择。

### 要了解的条件

- 您应该了解与系统时钟和时间戳记值的详细程度相关的条件。如果表包含 ROW CHANGE TIMESTAMP 列，那么在插入或更新操作后，新行在该数据库分区的该表中将具有唯一的 ROW CHANGE TIMESTAMP 值。
- 为了确保唯一性，生成的行时间戳记将始终增大，而无论系统时钟是否向后调整或者更新或插入数据的速度是否比时间戳记粒度要快。因此，与系统时间和 DB2 的 CURRENT TIMESTAMP 专用寄存器相比，ROW CHANGE TIMESTAMP 可能要超前一些。除非系统时钟完全不同步，或者数据库管理器每秒插入或更新的行数超过 1 百万行，否则此时间通常应非常接近实际时间。与 CURRENT TIMESTAMP 相比，此值也是在更新时对每行生成的，因此，它通常比 CURRENT TIMESTAMP 更接近实际时间。CURRENT TIMESTAMP 只对整个语句生成一次，并且完成该生成过程可能要花很长时间，这取决于受影响行的复杂度和数目。

### 在应用程序中启用乐观锁定

在应用程序中启用乐观锁定支持时需要执行一些步骤。

1. 在初始查询中，选择需要处理的每行的行标识（使用 第 227 页的『RID_BIT() 和 RID() 内置函数』）和行更改标记。
2. 释放行锁定，以便其他应用程序可以在表中执行选择、插入、更新和删除操作。
3. 通过在搜索条件中使用行标识和行更改标记对目标行执行搜索式 UPDATE 或 DELETE，并乐观地假定在执行原始 SELECT 语句后未锁定的行尚未更改。
4. 如果行已更改，那么更新操作将失败，并且应用程序逻辑必须处理该故障。例如，应用程序将重试选择和更新操作。

在运行上述步骤后:

- 如果应用程序执行的重试次数似乎比期望或需要的次数多，那么在表中添加行更改时间戳记列以确保只有对 RID_BIT 函数所标识的行进行的更改才会仅使行更改标记无效，而不会使同一数据页面上的其他活动无效。
- 要查看在给定时间范围内插入或更新的行，请创建或改变表以包含行更改时间戳记列。此列由数据库管理器自动维护，并且可以使用列名或 ROW CHANGE TIMESTAMP 表达式进行查询。
- 以下情况仅适用于行更改时间戳记列: 如果使用 IMPLICTLY HIDDEN 属性定义了列，那么当存在对表列的隐式引用时不会外部化该列。但是，始终可以在 SQL 语句中显式引用隐式隐藏的列。当在表中添加列会导致使用隐式列列表的现有应用程序失败时，这样做很有用。

## 表分区和数据组织方案

表分区是一种数据组织方案，它根据表的一个或多个分区列的值来将表数据分配到多个数据分区中。将给定表中的数据划分到多个存储对象中，这些存储对象可以位于不同表空间中。

有关表分区和数据组织方案的完整详细信息，请参阅 *《分区和集群指南》*。

## 创建表

数据库管理器控制对存储在表中的数据的更改和访问。可以使用 CREATE TABLE 语句创建表。可以使用复杂语句来定义表的所有属性和质量。但是，如果使用所有缺省值，那么用于创建表的语句非常简单。

```
CREATE TABLE <table name> (<column name> <data type> <column options>,
                           (<column name> <data type> <column options>, ...)
```

<table name> 不一定包含限定词。名称对系统目录中的所有表、视图和别名来说必须唯一。而且，名称还不能是 SYSIBM、SYSCAT、SYSFUN 或 SYSSTAT。

<column name> 对表中的列进行命名。不能限定此名称，并且它在该表的其他列中必须唯一。

对列存在的任何 <column options> 进一步定义该列的属性。选项包括用于防止列包含空值的 NOT NULL、用于 LOB 数据类型的特定选项、引用类型列的 SCOPE、对表的任何约束以及列的任何缺省值。有关更多信息，请参阅 CREATE TABLE 语句。

## 声明全局临时表

要在应用程序内创建全局临时表，可使用 DECLARE GLOBAL TEMPORARY TABLE 语句。

全局临时表（也称为用户定义的临时表）由处理数据库中的数据的应用程序使用。对数据进行处理产生的结果需要临时存储在表中。在创建全局临时表之前必须存在用户临时表空间。

**注**：全局临时表的描述并不出现在系统目录中，因此使其对于其他应用程序而言不是持久的，也不能与其他应用程序共享此表。当使用此表的应用程序终止或与数据库断开连接时，此表中的数据被删除，此表被隐式删除。
全局临时表不支持:

- LOB 类型列（或基于 LOB 的单值类型列）
- 用户定义的类型列
- LONG VARCHAR 列
- XML 列

### 示例

```
DECLARE GLOBAL TEMPORARY TABLE gbl_temp
         LIKE empltabl
         ON COMMIT DELETE ROWS
         NOT LOGGED
         IN usr_tbsp
```

此语句创建一个名为 gbl_temp 的全局临时表。定义此表所使用的列的名称和描述与 empltabl 的列的名称和描述完全相同。隐式定义只包括列名、数据类型、可空性特征和列缺省值属性。未定义所有其他列属性，包括唯一约束、外键约束、触发器和索引。执行 COMMIT 操作时，如果未对该表打开 WITH HOLD 游标，那么该表中的所有数据都被删除。不记录对用户临时表所作的更改。全局临时表被放在指定的用户临时表空间中。此表空间必须存在，否则此表的声明将失败。

如果创建此表时指定了 ROLLBACK 或 ROLLBACK TO SAVEPOINT，那么可以指定删除表中的所有行（DELETE ROWS，这是缺省情况），或者可以指定保留表中的各行（PRESERVE ROWS）。

在应用程序与数据库断开连接时，此表将被隐式删除。

## 创建类似于现有表的表

在发出指定了 ATTACH PARTITION 子句的 ALTER TABLE 语句时，如果目标表特征与源表特征之间的匹配不充分，那么有必要创建新的源表。在创建新的源表之前，可以尝试更正现有源表与目标表之间的不匹配情况。

要创建表，语句授权标识拥有的特权必须至少包括下列其中一项权限和特权:

- 对数据库的 CREATETAB 权限、对表空间的 USE 特权以及下列其中一项权限或特权:
  - 对数据库的 IMPLICIT_SCHEMA 权限（如果该表的隐式或显式模式名不存在）
  - 对模式的 CREATEIN 特权（如果该表的模式名引用现有模式）
- SYSADM 或 DBADM 权限

如果尝试更正不匹配情况失败，将返回 SQL20408N 或 SQL20307N 错误。

要创建新的源表:

1. 使用 db2look 命令来生成 CREATE TABLE 语句，以创建与目标表完全相同的表:

   ```
   db2look -d <source database name> -t <source table name> -e
   ```

2. 从 db2look 的输出中除去 partitioning 子句，将创建的表的名称更改为新名称（在本示例中，新名称是 sourceC）。
3. 接着，使用 LOAD FROM CURSOR 命令，将原始源表中的所有数据装入到新创建的源表 sourceC 中:

   ```
   DECLARE mycurs CURSOR FOR SELECT * FROM source
   LOAD FROM mycurs OF CURSOR REPLACE INTO sourceC
   ```

   如果此命令由于原始数据与表 sourceC 的定义不兼容而失败，在将原始表中的数据传送到 sourceC 时就必须对其进行变换。
4. 在成功地将数据复制到 sourceC 后，提交 ALTER TABLE target ...ATTACH sourceC 语句。

## 为登台数据创建表

*登台表*允许对延迟式具体化查询表的增量维护支持。登台表收集需要应用于具体化查询表以使其与基础表的内容同步的更改。使用登台表消除了在请求对具体化查询表的即时刷新时由于立即维护内容而引起的高锁定争用。另外，每次执行 REFRESH TABLE 时，不再需要完全重新生成具体化查询表。

在改进复杂查询的响应时间方面，具体化查询表是非常有效的方法，特别是针对可能需要下列某些操作的查询:

- 基于一个或多个维的聚集数据
- 连接和聚集涉及一组表的数据
- 经常访问的数据的子集中的数据
- 在分区数据库环境中，表或表的一部分中重新分区的数据

以下是关于登台表的一些关键限制:

1. 用来定义具体化查询表（为这个具体化查询表创建了登台表）的查询必须是可增量维护的；即，它必须与带有即时刷新选项的具体化查询表遵守相同的规则。
2. 只有延迟式刷新才能有支持登台表。查询还定义与登台表相关联的具体化查询表。具体化查询表必须定义为 REFRESH DEFERRED。
3. 使用登台表进行刷新时，仅支持刷新至当前时间点。
4. 不支持分区层次结构表和分区类型表。（分区表是这样的表: 根据 CREATE TABLE 语句的 PARTITION BY 子句中指定的内容，数据被划分到多个存储对象中。）

不能使用不一致、不完整或处于暂挂状态的登台表来增量刷新相关联的具体化查询表，除非执行了某些其他操作。这些操作将使登台表的内容与其相关联的具体化查询表及其基础表保持一致，并使登台表脱离暂挂状态。刷新具体化查询表之后，其登台表的内容会被清除并将登台表设置为正常状态。还可以使用带有相应选项的 SET INTEGRITY 语句有目的地修剪登台表。修剪会将登台表更改为不一致状态。例如，下列语句强制修剪称为 STAGTAB1 的登台表:

```
SET INTEGRITY FOR STAGTAB1 PRUNE;
```

创建登台表时，会将它置于暂挂状态并会出现一个指示符，显示该表对于基础表的内容及相关联的具体化查询表是不一致或不完整的。需要使登台表脱离暂挂状态和不一致状态以便开始收集对其基础表的更改。处于暂挂状态时，试图对任何登台表的基础表进行修改都将失败，试图刷新相关联的具体化查询表也会失败。

有几种方法可使登台表脱离暂挂状态；例如:

- SET INTEGRITY FOR <staging table name> STAGING IMMEDIATE UNCHECKED
- SET INTEGRITY FOR <staging table name> IMMEDIATE CHECKED

# 修改表

本节提供有关如何修改表的主题。

## 改变具体化查询表属性

在遵守某些限制的前提下，可将具体化查询表更改为常规表或将常规表更改为具体化查询表。不能更改其他表类型；只能更改常规表和具体化查询表。例如，不能将复制具体化查询表更改为常规表，反之亦然。

一旦将正规表改变为具体化查询表，该表便处于设置完整性暂挂状态。当使用此方法进行改变时，具体化查询表定义中的全查询 定义必须与原始表定义相匹配，即:

- 列数必须相同。
- 列名和位置必须匹配。
- 数据类型必须完全相同。

如果具体化查询表是对原始表定义的，那么不能将原始表本身改变为具体化查询表。如果原始表有触发器、检查约束、引用约束或定义的唯一索引，那么不能将其改变为具体化查询表。如果改变表属性来定义具体化查询表，那么不允许在同一 ALTER TABLE 语句中以任何其他方法改变该表。

在将常规表改变为具体化查询表时，具体化查询表定义的全查询不能直接引用原始表或通过视图、别名或具体化查询表间接引用原始表。

要将具体化查询表更改为常规表，使用以下命令:

```
ALTER TABLE sumtable
      SET SUMMARY AS DEFINITION ONLY
```

要将常规表更改为具体化查询表，使用以下命令:

```
 ALTER TABLE regtable
      SET SUMMARY AS <fullselect>
```

将常规表改变为具体化查询表时，有关全查询的限制与使用 CREATE SUMMARY TABLE 语句创建摘要表时的限制非常相似。

## 刷新具体化查询表中的数据

可通过使用 REFRESH TABLE 语句来刷新一个或多个具体化查询表中的数据。此语句可嵌入应用程序中，或可动态执行。要使用此语句，必须具有 SYSADM 或 DBADM 权限，或对要刷新的表具有 CONTROL 特权。

以下示例显示如何刷新具体化查询表中的数据:

```
REFRESH TABLE SUMTAB1
```

## 更改列属性

使用 ALTER TABLE 语句来更改列属性，例如，可空性、LOB 选项、作用域、约束、压缩属性以及数据类型等等。有关完整的详细信息，请参阅 ALTER TABLE 语句。

要改变表，您必须对要改变的表至少具有下列其中一种特权:

- ALTER 特权
- CONTROL 特权
- SYSADM 或 DBADM 权限
- 对表模式的 ALTERIN 特权

要更改现有列的定义、在更改表列时编辑和测试 SQL 或者在更改表列时验证相关对象，您必须具有 DBADM 权限。

例如，在命令行中输入:

```
ALTER TABLE EMPLOYEE
        ALTER COLUMN WORKDEPT
           SET DEFAULT '123'
```

### 添加和删除列

要将列添加至现有表，或者从现有表中删除列，可分别使用带有 ADD COLUMN 或 DROP COLUMN 子句的 ALTER TABLE 语句。表不能是类型表。

对于表中的所有现有列，新列的值将设置为其缺省值。新列是表中的最后一列；也就是说，如果最初有 *n* 列，那么添加的列将是第 *n*+1 列。添加新列不能使所有列的总字节计数超过行大小限制。

要添加列，请发出以下语句:

```
ALTER TABLE SALES
  ADD COLUMN SOLD_QTY
  SMALLINT NOT NULL DEFAULT 0
```

要删除列，请发出以下语句:

```
ALTER TABLE SALES
  DROP COLUMN SOLD_QTY
```

### 修改 DEFAULT 子句列定义

DEFAULT 子句用于在以下事件中为列提供缺省值: INSERT 中未提供值或者值指定为 INSERT 或 UPDATE 中的 DEFAULT。如果在 DEFAULT 关键字后未指定特定缺省值，那么缺省值将取决于数据类型。如果列定义为 XML 或结构化类型，那么不能指定 DEFAULT 子句。

在列定义中省略 DEFAULT 会导致将空值用作该列的缺省值，如第 213 页的『缺省列和数据类型定义』中所述。

可以使用 DEFAULT 关键字指定特定类型的值，请参阅 ALTER TABLE 语句。

## 修改列的生成或标识属性

可以使用 ALTER TABLE 语句中的 ALTER COLUMN 子句，在表中添加或删除列的生成或标识属性。

可执行下列其中一项操作:

- 使用现有非生成列时，可以添加生成的表达式属性。修改的列则成为生成列。
- 使用现有生成列时，可以删除生成的表达式属性。修改的列则成为正常的非生成列。
- 使用现有非标识列时，可以添加标识属性。修改的列则成为标识列。
- 使用现有标识列时，可以删除标识属性。修改的列则成为正常的非生成、非标识列。
- 使用现有标识列时，可以将标识列从 GENERATED ALWAYS 改变为 GENERATED BY DEFAULT。反之亦然，即，可将标识列从 GENERATED BY DEFAULT 改变为 GENERATED ALWAYS。只有使用标识列时，才能执行此操作。
- 可以从用户定义的缺省列中删除缺省属性。执行此操作时，新缺省值为空。
- 可以删除缺省值、标识或生成属性，然后在相同的 ALTER COLUMN 语句中设置新的缺省值、标识或生成属性。
- 对于 CREATE TABLE 和 ALTER TABLE 语句，“ALWAYS”关键字在生成列子句中是可选的。这意味着 GENERATED ALWAYS 等价于 GENERATED。

## 修改列定义

使用 ALTER TABLE 语句来删除列或更改其类型和属性。例如，可以增加现有 VARCHAR or VARGRAPHIC 列的长度。字符数可增加到与所用的页大小相关的一个值。

要修改与列关联的缺省值，在定义了新缺省值后，将对任何后续 SQL 操作中指示使用此缺省值的列使用新值。新值必须遵守赋值规则，且受到与 CREATE TABLE 语句下记录的限制相同的限制。

**注**: 生成列无法通过该语句来改变其缺省值。

在使用 SQL 更改这些表属性时，不再需要删除表然后重新创建它，不然如果存在对象依赖关系，处理起来就会很复杂，需要花费很多时间。

要使用命令行来修改现有表列的长度和类型，请输入:

```
ALTER TABLE <table_name>
      ALTER COLUMN <column_name>
 <modification_type>
```

例如，要将一列增加到 4000 个字符，使用类似于以下的语句:

```
ALTER TABLE t1
      ALTER COLUMN colnam1
      SET DATA TYPE VARCHAR(4000)
```

在另一个示例中，要允许一列具有新的 VARGRAPHIC 值，使用类似以下内容的语句:

```
ALTER TABLE t1
      ALTER COLUMN colnam2
      SET DATA TYPE VARGRAPHIC(2000)
```

不能改变类型表的列。但是，可将一个作用域添加到尚未定义作用域的现有的引用类型列中。例如:

```
ALTER TABLE t1
      ALTER COLUMN colnamt1
      ADD SCOPE typtab1
```

要使用命令行来修改现有表列的缺省值，请输入:

```
ALTER TABLE <table_name>
      ALTER COLUMN <column_name>
SET DEFAULT 'new_default_value'
```

例如，要更改列的缺省值，请使用以下类似的语句:

```
ALTER TABLE t1
      ALTER COLUMN colnam1
      SET DEFAULT '123'
```

## 重命名表

可以使用 RENAME 语句来重命名现有表。

重命名表时，源表不能在任何现有定义（视图或具体化查询表）、触发器、SQL 函数或约束中引用。它也不能具有任何生成列（标识列除外），或者不能是父表或从属表。目录条目将更新以反映新表名。有关更多信息和示例，请参阅 RENAME 语句。

要更改现有列的定义，请参阅第 234 页的『更改列属性』和 ALTER TABLE 语句。

## 恢复不可用摘要表

撤销基础表的 SELECT 特权将会导致摘要表变得*不可用*。

下列步骤可帮助您恢复不可用摘要表:

- 确定最初用于创建该摘要表的语句。可以从 SYSCAT.VIEW 目录视图的 TEXT 列获取此信息。
- 使用 CREATE SUMMARY 语句并使用相同的摘要表名和相同的定义，来重新创建该摘要表。
- 使用 GRANT 语句重新授予先前在该摘要表上授予的所有特权。（注意，在不可用摘要表上授予的所有特权都被撤销。）

若不希望恢复不可用摘要表，可以使用 DROP TABLE 语句显式地删除它，或者可以使用相同的名称但是不同的定义来创建新的摘要表。

不可用摘要表只在 SYSCAT.TABLES 和 SYSCAT.VIEWS 目录视图中具有条目；在 SYSCAT.TABDEP、SYSCAT.TABAUTH、SYSCAT.COLUMNS 和 SYSCAT.COLAUTH 目录视图中的所有条目已被除去。

## 查看表定义

可以使用 SYSCAT.COLUMNS 目录视图来查看表定义。每行都表示对表、视图或昵称定义的一列。要查看列中的数据，请使用 SELECT 语句。

## 表或视图别名

*别名*是表或视图的另一个名称。如果*可以*引用现有表或视图，那么可以使用别名来引用该表或视图。

并不是所有上下文中都可以使用别名；例如，在检查约束的检查条件中不能使用别名。别名不能引用已声明的临时表。

与表或视图一样，可以创建和删除别名，并且它可以具有关联的注释。但是，与表不同的是，别名可以在称为*链接*的过程中相互引用。别名是公共引用的名称，因此使用别名时不需要特殊权限或特权。但是，访问别名所引用的表或视图需要与这些对象关联的权限。

存在其他类型的别名，例如，数据库和网络别名。还可以为表示联合系统上的数据表或视图的*昵称*创建别名。

## 删除表

可以使用 DROP TABLE 语句删除表。当删除一个表时，也会删除 SYSCAT.TABLES 系统目录中包含有关该表的信息的那一行，并会影响从属于该表的任何其他对象。

例如:

- 会删除所有的列名。
- 会删除基于该表的任何列创建的索引。
- 将基于该表的所有视图标记为不可用。
- 对删除的表和从属视图的所有特权被隐式撤销。
- 会删除在其中该表为父表或从属表的所有引用约束。
- 从属于删除的表的所有程序包和高速缓存的动态 SQL 和 XQuery 语句被标记为无效，且该状态会保持至重新创建了从属对象为止。这包括这样一些程序包，它们从属于将被删除的层次结构中子表上的任何超表。
- 其引用的作用域为删除的表的任何引用列变为“无作用域”。
- 因为可以取消定义别名，所以该表上的别名定义不受影响
- 将从属于该删除的表的所有触发器标记为不可用。

要使用命令行来删除表，请输入:

```
DROP TABLE <table_name>
```

以下语句删除 DEPARTMENT 表:

```
DROP TABLE DEPARTMENT
```

如果一个表有子表，那么不能删除该表。但是，可用单个 DROP TABLE HIERARCHY 语句删除一个表 e 中的所有表，如以下示例所示:

```
DROP TABLE HIERARCHY person
```

DROP TABLE HIERARCHY 语句必须命名要删除的层次结构的根表。

删除表层次结构与删除特定的表之间有一些差别:

- DROP TABLE HIERARCHY 不会激活个别的 DROP 表语句将激活删除触发器。例如，删除个别子表将激活其超表上的删除触发器。

- DROP TABLE HIERARCHY 不为删除的表的个别行建立日志条目。而是将该层次结构的删除作为单个事件记录。

## 删除具体化查询表或登台表

不能改变但是可以删除具体化查询表或登台表。所有引用该表的索引、主键、外键和检查约束均被删除。所有引用该表的视图和触发器均变得不可用。从属于任何删除的对象的程序包或被标记为不可用的程序包均将无效。

要使用命令行来删除具体化查询表或登台表，请输入:

```
DROP TABLE <table_name>
```

以下语句将删除具体化查询表 XT:

```
DROP TABLE XT
```

可使用 DROP TABLE 语句显式删除具体化查询表，如果删除了任何一个基础表，那么可能会隐式删除该具体化查询表。

可使用 DROP TABLE 语句显式删除登台表，如果删除了其关联的具体化查询表，那么可能会隐式删除该登台表。

# 表方案和示例

本节提供表方案和示例。

## 方案: 乐观锁定和基于时间的检测

提供了三个方案来说明如何在使用和不使用基于时间的检测以及使用和不使用隐式隐藏的列的情况下在应用程序中启用和实施乐观锁定。

### 方案: 在应用程序中使用乐观锁定

此方案说明如何在应用程序中实施乐观锁定，它包括六种不同的情况。

请考虑针对乐观锁定设计并已启用乐观锁定的应用程序中的下列事件顺序:

```
SELECT QUANTITY, row change token FOR STOCK, RID_BIT(STOCK)
 INTO :h_quantity, :h_rct, :h_rid
 FROM STOCK WHERE PARTNUM = 3500
```

在此方案中，应用程序逻辑读取每行。由于已按第 230 页的『在应用程序中启用乐观锁定』中所描述的对此应用程序启用乐观锁定，所以选择列表包括保存在 :h_rid 主变量中的 RID_BIT() 值和保存在 :h_rct 主变量中的行更改标记值。

在启用了乐观锁定的情况下，应用程序乐观地假定更新或删除操作的任何目标行都保持不变，即使未通过锁定来保护它们亦如此。为了提高数据库并行性，应用程序使用下列其中一种方法除去行锁定:

- 落实工作单元，在这种情况下行锁定将被除去
- 使用 WITH RELEASE 子句关闭游标，在这种情况下行锁定将被除去
- 使用较低的隔离级别:
  - 游标稳定性（CS），在这种情况下，在游标访存到下一行或结果表末尾后行未锁定。

– 未落实的读（UR），在这种情况下，任何未落实的数据将具有新的（未落实）行更改标记值。如果回滚未落实的数据，那么已落实的旧行更改标记将是另一个值。

  **注**: 假定通常不回滚更新，使用 UR 将允许最大并行性。

- 断开与数据库的连接，因此释放应用程序的所有 DB2 服务器资源。（.NET 应用程序通常使用此方式）。

应用程序处理这些行并决定要乐观地更新其中一行:

```
UPDATE STOCK SET QUANTITY = QUANTITY – 1
 WHERE row change token FOR STOCK = :h_rct AND
 RID_BIT(STOCK) = :h_rid
```

UPDATE 语句更新上面显示的 SELECT 语句中标识的行。

搜索式 UPDATE 谓词计划用作对表的直接访存:

```
RID_BIT(STOCK) = :h_rid
```

直接访存是一种非常有效的访问方案，DB2 优化器很容易就可以估计其成本。如果 RID_BIT() 谓词未找到行，那么表示行已删除并且更新操作由于未找到行而失败。

假定 RID_BIT() 谓词找到了行，那么在行更改标记未更改时，行更改标记谓词 FOR STOCK = :h_rct 将找到该行。如果在选择操作后行更改标记已更改，那么搜索式 UPDATE 将由于未找到行而失败。

表 49 列示了在启用乐观锁定后可能出现的情况。

*表 49. 启用乐观锁定后可能出现的情况*

| 情况标识 | 操作 | 结果 |
|---|---|---|
| **情况 1** | 表中定义了行更改时间戳记列，并且其他应用程序未更改行。 | 由于行更改标记谓词成功找到 :h_rid 所标识的行，所以更新操作成功。 |
| **情况 2** | 表中定义了行更改时间戳记。另一个应用程序在选择操作后在更新（落实）操作前更新了行，从而更新了行更改时间戳记列。 | 行更改标记谓词未能将在选择操作时根据行中的时间戳记生成的标记与行中当前时间戳记的标记值进行比较。因此 UPDATE 语句找不到行。 |
| **情况 3** | 表中定义了行更改时间戳记。另一个应用程序更新了行，因此行具有新的行更改标记。此应用程序在隔离级别 UR 选择行，并获取未落实的新行更改标记。 | 此应用程序运行 UPDATE 语句，这将锁定等待，直到其他应用程序释放其行锁定为止。如果其他应用程序使用新标记落实更改，那么行更改标记谓词将成功，因此 UPDATE 语句成功。如果其他应用程序回滚到旧标记，那么行更改标记谓词将失败，因此 UPDATE 语句找不到行。 |
| **情况 4** | 表中未定义任何行更改时间戳记列。在选择操作后在更新操作前，在同一页面上更新、删除或插入了另一行。 | 由于该页面上所有行的行更改标记值已更改，所以行更改标记谓词未能比较标记，因此 UPDATE 语句找不到行，即使行实际上并未更改亦如此。<br><br>如果添加了行更改时间戳记列，那么此被动错误信息情况不会导致 UPDATE 语句失败。 |

*表 49. 启用乐观锁定后可能出现的情况 (续)*

| 情况标识 | 操作 | 结果 |
| --- | --- | --- |
| **情况 5** | 改变了表以便包含行更改时间戳记列，并且在改变操作后选择中返回的行未修改。另一个应用程序更新该行，从而在此过程中将具有当前时间戳记的行更改时间戳记列添加至该行。 | 行更改标记谓词未能将先前生成的标记与根据行更改时间戳记列创建的标记值进行比较，因此 UPDATE 语句找不到行。由于感兴趣的行实际上已更改，因此这不是被动错误信息情况。 |
| **情况 6** | 在选择操作后在更新操作前重组了表。:h_rid 所标识的行标识找不到行，或者它包含具有另一个标记的行，因此更新操作失败。这是无法避免的被动错误信息情况，即使行中存在行更改时间戳记列也无法避免这种情况。 | 行本身未被重组操作更新，但重组后谓词的 RID_BIT 部分无法标识原始行。 |

## 方案: 使用隐式隐藏的列的乐观锁定

下列方案说明如何在应用程序中使用隐式隐藏的列实施乐观锁定，隐式隐藏的列是使用 IMPLICITLY HIDDEN 属性定义的列。

对于这些方案，假定定义的 SALARY_INFO 表包含三列，并且第一列是隐式隐藏的 ROW CHANGE TIMESTAMP 列，始终会生成该列的值。

**情况 1**:

在以下语句中，列列表中显式引用隐式隐藏的列，并且在 VALUES 子句中提供了该列的值:

```
INSERT INTO SALARY_INFO (UPDATE_TIME, LEVEL, SALARY)
  VALUES (DEFAULT, 2, 30000)
```

**情况 2**:

以下 INSERT 语句使用隐式列列表。隐式列列表不包含隐式隐藏的列，因此，VALUES 子句只包含其他两列的值:

```
INSERT INTO SALARY_INFO
  VALUES (2, 30000)
```

在本示例中，必须将列 UPDATE_TIME 定义为具有缺省值，并将该缺省值用于插入的行。

**情况 3**:

在以下语句中，选择列表中显式引用隐式隐藏的列，并且该列的值出现在结果集中:

```
SELECT UPDATE_TIME, LEVEL, SALARY FROM SALARY_INFO
  WHERE LEVEL = 2

  UPDATE_TIME        LEVEL    SALARY
  -------------------------- ----------- -----------
  2006-11-28-10.43.27.560841      2    30000
```

**情况 4**:

在以下语句中，列列表是通过使用 * 表示法隐式生成的，并且隐式隐藏的列不出现在结果集中:

```
SELECT * FROM SALARY_INFO
  WHERE LEVEL = 2

  LEVEL    SALARY
  ---------- -----------
       2    30000
```

**情况 5**:

在以下语句中，列列表是通过使用 * 表示法隐式生成的，并且通过使用 ROW CHANGE TIMESTAMP FOR 表达式隐式隐藏的列值也会显示:

```
   SELECT ROW CHANGE TIMESTAMP FOR SALARY_INFO AS ROW_CHANGE_STAMP,
SALARY_INFO.*
FROM SALARY_INFO WHERE LEVEL = 2
```

结果表将类似于情况 3（列 UPDATE_TIME 将为 ROW_CHANGE_STAMP）。

## 方案: 基于时间的更新检测

此方案说明如何在应用程序中按时间戳记使用更新检测来实施乐观锁定，它包括三种不同的情况。

在此方案中，应用程序选择在过去 30 天内更改的所有行。

```
SELECT * FROM TAB WHERE
  ROW CHANGE TIMESTAMP FOR TAB <=
  CURRENT TIMESTAMP AND
  ROW CHANGE TIMESTAMP FOR TAB >=
  CURRENT TIMESTAMP - 30 days;
```

**情况 1**:

表中未定义任何行更改时间戳记列。语句失败，并发出 SQL20431N。仅定义了行更改时间戳记列的表支持此 SQL 表达式。

**注**: 这种情况在 z/OS 上起作用。

**情况 2**:

在创建表时定义了行更改时间戳记列。

```
CREATE TABLE TAB ( ..., RCT TIMESTAMP NOT NULL
                        GENERATED ALWAYS
                        FOR EACH ROW ON UPDATE AS
                        ROW CHANGE TIMESTAMP)
```

此语句返回在过去 30 天内插入或更新的所有行。

**情况 3**:

在过去 30 天中的某个时刻使用 ALTER TABLE 语句将行更改时间戳记列添加至表:

```
ALTER TABLE TAB ADD COLUMN RCT TIMESTAMP NOT NULL
                               GENERATED ALWAYS
                               FOR EACH ROW ON UPDATE AS
                               ROW CHANGE TIMESTAMP
```

此语句返回表中的所有行。在执行 ALTER TABLE 语句之后尚未修改的任何行将使用 ALTER TABLE 语句本身的时间戳记缺省值，而在此之后已修改的所有其他行将具有唯一时间戳记。

# 第 12 章 约束

在任何业务中，数据通常必须符合特定限制或规则。例如，职员编号必须是唯一的。数据库管理器提供了*约束*作为强制实施这种规则的方法。

提供了下列类型的约束:

- NOT NULL 约束
- 唯一（或唯一键）约束
- 主键约束
- 外键（或引用完整性）约束
- （表）检查约束
- 参考约束

约束只与表关联，它们是在创建表的过程中定义的（使用 CREATE TABLE 语句）或者是在创建表后添加至表定义的（使用 ALTER TABLE 语句）。可以使用 ALTER TABLE 语句来修改约束。在大多数情况下，随时都可以删除现有约束；此操作不会影响表结构和存储在表中的数据。

**注**: 唯一约束和主键约束只与表对象关联，它们通常是使用一个或多个唯一或主键索引强制执行的。

## 约束的类型

*约束*是用于优化的规则。

有五种类型的约束:

- *NOT NULL 约束*是这样一种规则，它防止在表的一列或多列中输入空值。
- *唯一约束*（也称为*唯一键约束*）是这样一种规则，它禁止表的一列或多列中出现重复值。唯一键和主键是受支持的唯一约束。例如，可对供应商表中的供应商标识定义唯一约束以确保不会对两个供应商指定同一供应商标识。
- *主键约束*是与唯一约束具有相同属性的一列或列的组合。可使用主键和外键约束来定义表之间的关系。
- *外键约束*（也称为*引用约束*或*引用完整性约束*）是关于一个或多个表中的一列或多列中的值的一种逻辑规则。例如，一组表共享关于公司的供应商的信息。供应商的名称有时可能会更改。可定义一个引用约束，声明表中的供应商的标识必须与供应商信息中的供应商标识相匹配。此约束会阻止可能导致丢失供应商信息的插入、更新或删除操作。
- *（表）检查约束*（简称为*检查约束*）对添加至特定表的数据设置限制。例如，表检查约束可确保每当在包含个人信息的表中添加或更新薪水数据时，职员的薪水级别至少为 $20,000。

*参考约束*是不由数据库管理器强制执行的某种类型的约束的属性。

© Copyright IBM Corp. 1993, 2010

## NOT NULL 约束

NOT NULL 约束防止在列中输入空值。

数据库中使用空值来表示未知状态。缺省情况下，随数据库管理器一起提供的所有内置数据类型都支持空值的存在。但是，一些业务规则可能要求必须始终提供值（例如，要求每位职员提供紧急联系人信息）。NOT NULL 约束用于确保决不会为给定表列指定空值。为特定列定义 NOT NULL 约束后，尝试在该列中放入空值的任何插入或更新操作将失败。

因为约束仅适用于特定表，所以它们通常是在创建表的过程中与表属性一起定义的。以下 CREATE TABLE 语句显示了如何为特定列定义 NOT NULL 约束:

```
CREATE TABLE EMPLOYEES    (. . .
                           EMERGENCY_PHONE    CHAR(14)  NOT NULL,
                           . . .
                           );
```

## 唯一约束

*唯一约束*确保一组列中的值对于表中的所有行都是唯一的，且不为空。在唯一约束中指定的列必须定义为 NOT NULL。数据库管理器使用唯一索引在对唯一约束的各列进行更改时强制键的唯一性。

可在 CREATE TABLE 或 ALTER TABLE 语句中使用 UNIQUE 子句定义唯一约束。例如，DEPARTMENT 表中的典型唯一约束可以是: 部门号是唯一的，且不为空。

图 21 显示了当表存在唯一约束时，阻止将重复的记录添加到该表。

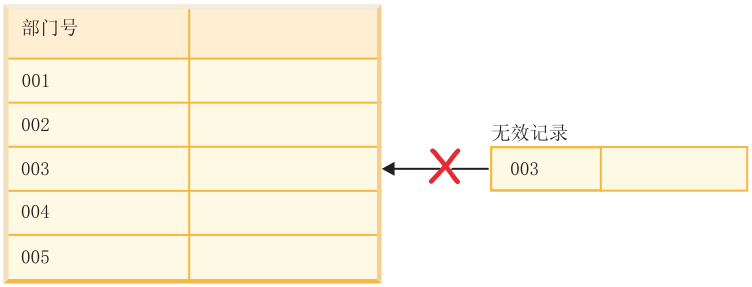



*图 21. 唯一约束防止出现重复数据*

数据库管理器在插入和更新操作期间强制执行此约束，以确保数据完整性。

表可以有任意数目的唯一约束，最多将一个唯一约束定义为主键。对于同一组列，表不能有多个唯一约束。

引用约束的外键引用的唯一约束称为*父键*。

- 当在 CREATE TABLE 语句中定义唯一约束时，唯一索引是由数据库管理器自动创建的，且被指定为主索引或系统所需的唯一索引。

- 当在 ALTER TABLE 语句中定义唯一约束且同一组列存在索引时，该索引被指定为唯一的且是系统所需的。如果这样的索引不存在，数据库管理器会自动创建唯一索引，并将其指定为主索引或系统所需的唯一索引。

**注:** 定义唯一约束与创建唯一索引是有区别的。尽管都强制唯一性，但唯一索引允许可空列，且通常不能用作父键。

## 主键约束

可以使用主键约束和外键约束来定义表之间的关系。

主键是与唯一约束具有相同属性的一个列或列的组合。因为主键用来标识表中的一行，所以它必须是唯一的，并且必须具有 NOT NULL 属性。一个表不能有多个主键，但可以有多个唯一键。主键是可选的，可以在创建或改变表时定义。当导出或重组数据时，主键可以对数据进行排序，所以它们也是有益的。

## （表）检查约束

*检查约束*（也称为*表检查约束*）是这样一种数据库规则，它指定表中每行的一列或多列中允许使用的值。指定检查约束是通过限制格式的搜索条件完成的。

## 外键（引用）约束

*外键约束*（也称为*引用约束*或*引用完整性约束*）使您能够定义表间以及表内必需的关系。

例如，典型的外键约束可能规定 EMPLOYEE 表中的每个职员必须是一个现有部门的成员，该部门在 DEPARTMENT 表中定义。

*引用完整性*是数据库的一种状态，在该状态中，所有外键的所有值都有效。*外键*是表中的一列或一组列，它的值需要与其父表的行的至少一个主键或唯一键值相匹配。*引用约束*是这样一种规则，仅当满足下列其中一个条件时，外键的值才有效:

- 它们作为父键的值出现。
- 外键的某些组成部分为空。

要建立此关系，应将 EMPLOYEE 表中的部门号定义成外键，并将 DEPARTMENT 表中的部门号定义成主键。

第 246 页的图 22 显示了当两个表之间存在外键约束时，如何阻止将具有无效键的记录添加至表:

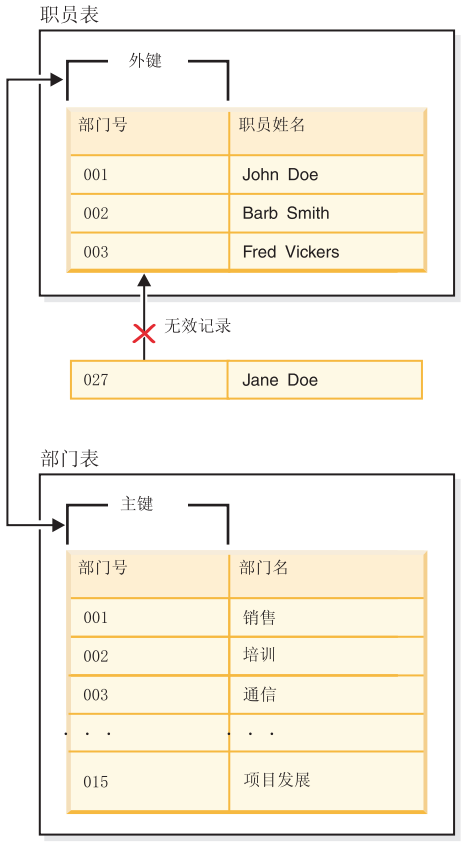



*图 22. 外键和主键约束*

包含父键的表称为引用约束的父表，包含外键的表被认为是该表的从属表。

可以在 CREATE TABLE 语句或 ALTER TABLE 语句中定义引用约束。引用约束由数据库管理器在执行 INSERT、UPDATE、DELETE、ALTER TABLE、MERGE、ADD CONSTRAINT 和 SET INTEGRITY 语句时强制实施。

引用完整性规则涉及下列术语:

*表 50. 引用完整性术语*

| 概念 | 术语 |
|---|---|
| 父键 | 引用约束的主键或唯一键。 |
| 父行 | 具有至少一个从属行的行。 |
| 父表 | 包含引用约束的父键的表。表可在任意数目的引用约束中充当父表。在引用约束中充当父表的表还可以是引用约束中的从属表。 |
| 从属表 | 在其定义中包含至少一个引用约束的表。表可在任意数目的引用约束中充当从属表。在引用约束中充当从属表的表还可以是引用约束中的父表。 |
| 派生表 | 作为表 T 的后代的表（如果它是 T 的从属表或是 T 的从属表的后代）。 |
| 从属行 | 具有至少一个父行的行。 |
| 派生行 | 作为行 r 的后代的行（如果它是 r 的从属行或是 r 的从属行的后代）。 |
| 引用循环 | 引用约束的集合，该集合中的每个表都是它自身的后代。 |
| 自引用表 | 在同一引用约束中既充当父表又充当从属表的表。该约束称为*自引用约束*。 |

*表 50. 引用完整性术语 (续)*

| 概念 | 术语 |
| --- | --- |
| 自引用行 | 一个作为它自己的父代的行。 |

引用约束的目的是保证表关系得到维护并遵循数据输入规则。这意味着只要引用约束有效，数据库管理器就保证对于子表中其外键列中具有非空值的每行，相应父表中都存在一个其父键中具有匹配值的行。

当 SQL 操作尝试更改数据的方式导致引用完整性受到影响时，可能是违反了外键（或引用）约束。数据库管理器通过强制执行与每个引用约束关联的一组规则来处理这类情况。这组规则包括:

- 插入规则
- 更新规则
- 删除规则

当 SQL 操作尝试更改数据的方式导致引用完整性受到影响时，可能是违反了引用约束。例如,

- 插入操作可能尝试将一个数据行插入到子表中，该行的外键列中的值与相应父表的父键中的值不匹配。
- 更新操作可能尝试将子表的外键列中的值更改为一个在相应父表的父键中没有匹配值的值。
- 更新操作可能尝试将父表的父键中的值更改为一个在子表的外键列中没有匹配值的值。
- 删除操作可能尝试从父表中除去在子表的外键列中具有匹配值的记录。

数据库管理器通过强制执行与每个引用约束关联的一组规则来处理这类情况。这组规则包括:

- 插入规则
- 更新规则
- 删除规则

## 插入规则

引用约束的插入规则为: 外键的非空插入值必须与父表的父键的某些值相匹配。如果组合外键的值的任何组成部分为空，那么该值为空。指定外键时，此规则是隐式的。

## 更新规则

引用约束的更新规则是在定义引用约束时指定的。选项有 NO ACTION 和 RESTRICT。在更新父表的某行或从属表的某行时应用更新规则。

如果是父行，更新父键的某列中的值时，下列规则适用:

- 如果从属表中的任何行与该键的原始值相匹配，在更新规则为 RESTRICT 的情况下，会拒绝更新。
- 如果在更新语句完成时从属表中的任意行没有相应的父键（排除后触发器），当更新规则为 NO ACTION 时，会拒绝更新。

如果更新规则为 RESTRICT，并且存在一个或多个从属行，那么父代唯一键的值不能更改。但是，如果更新规则为 NO ACTION，并且更新语句完成时每个子代都有父键，那么父代唯一键可以更新。外键的非空更新值必须等于关系的父表的主键值。

而且，将 NO ACTION 或 RESTRICT 用作引用约束的更新规则将确定何时强制执行约束。更新规则 RESTRICT 将在所有其他约束之前（包括修改 CASCADE 或 SET NULL 之类的规则的引用约束）执行。更新规则 NO ACTION 将在其他引用约束之后强制执行。注意，返回的 SQLSTATE 根据更新规则是 RESTRICT 还是 NO ACTION 而有所不同。

如果是从属行，当指定外键时，NO ACTION 更新规则是隐式的。NO ACTION 意味着更新语句完成时，外键的非空更新值必须与父表的父键的某些值相匹配。

如果组合外键的值的任何组成部分为空，那么该值为空。

## 删除规则

引用约束的删除规则是在定义引用约束时指定的。选项有 NO ACTION、RESTRICT、CASCADE 或 SET NULL。仅当外键的某些列允许空值时，才能指定 SET NULL。

如果已标识表或已标识视图的基本表为父代，那么选择要删除的行在删除规则 RESTRICT 的关系中一定不能有任何从属项，并且 DELETE 一定不能级联至在删除规则 RESTRICT 的关系中具有从属的后代行。

如果 RESTRICT 删除规则未阻止删除操作，那么会删除所选行。从属于所选行的所有行也会受到影响:

- 在删除规则 SET NULL 的关系中充当从属项的所有行的外键的可空列将设置为空值。
- 在删除规则 CASCADE 的关系中充当从属项的所有行也会被删除，而上述规则将应用于这些行。

将检查删除规则 NO ACTION 以在强制执行其他引用约束后强制所有非空外键引用现有父行。

仅当删除父表的*一行*后，引用约束的删除规则才适用。更准确地说，仅当父表的*一行*成为删除或传播删除操作（下面将做出定义）的对象并且该行在引用约束的从属表中具有从属项时，此规则才适用。考虑这样一个示例，其中 P 是父表，D 是从属表，而 p 是充当删除或传播删除操作的对象的父行。删除规则的工作方式如下:

- 对于 RESTRICT 或 NO ACTION，发生错误，且不会删除任何行。
- 对于 CASCADE，删除操作会传播至表 D 中的 p 的从属项。
- 对于 SET NULL，表 D 中的 p 的每个从属项的外键的每个可空列被设置为空。

涉及针对 P 的删除操作的任何表都被认为是*删除连接*至 P。因此，如果一个表是 P 的从属项，或是 P 中的删除操作级联至的表的从属项，那么该表删除连接至表 P。

下列限制适用于删除连接关系:

- 如果一个表在多个表的引用循环中删除连接至它自己，那么该循环不能包含删除规则 RESTRICT 或 SET NULL。
- 一个表不能既是 CASCADE 关系中的从属表（自引用或者引用另一个表）又与删除规则 RESTRICT 或 SET NULL 具有自引用关系。

- 当一个表通过多种关系（这些关系具有重叠的外键）删除连接至另一个表时，这些关系必须具有相同的删除规则，并且任何这些关系都不能为 SET NULL。
- 当一个表通过多种关系（其中一种关系是使用删除规则 SET NULL 指定的）删除连接至另一个表时，此关系的外键定义不能包含任何分布键或 MDC 键列，也不能添加数据分区键列或 RCT 键列。
- 当两个表通过 CASCADE 关系删除连接至同一个表时，这两个表之间不能互相删除连接（其中删除连接路径以删除规则 RESTRICT 或 SET NULL 结束）。

## 参考约束

*参考约束*是一种约束属性，SQL 编译器可使用它来改善对数据的访问。参考约束不是由数据库管理器强制执行的，并且不用于数据的附加验证；它们用来提高查询性能。

参考约束是使用 CREATE TABLE 或 ALTER TABLE 语句定义的。首先添加引用完整性或检查约束，然后使约束属性与它们相关联以指定数据库管理器是否强制执行约束；以及是否将约束用于查询优化。

# 设计约束

在设计和创建约束时，最好使用正确标识不同类型的约束的命名约定。这对于诊断可能出现的错误来说特别重要。

可设计下列类型的约束:

- NOT NULL 约束
- 唯一约束
- 主键约束
- （表）检查约束
- 外键（引用）约束
- 信息约束

## 设计唯一约束

*唯一约束*确保在指定键中的每个值都是唯一的。一个表可以有任意多个唯一约束，且将一个唯一约束定义为主键。

**限制**

- 可能不能对子表定义唯一约束。
- 每个表只能有一个主键。

可在 CREATE TABLE 或 ALTER TABLE 语句中使用 UNIQUE 子句来定义唯一约束。唯一键可以由多个列组成。在一个表上允许多个唯一约束。

一旦建立了该约束，当 INSERT 或 UPDATE 语句修改表中的数据时，数据库管理器会自动实现该唯一约束。唯一约束通过唯一索引来实现。

当在 ALTER TABLE 语句中定义唯一约束且在该唯一键的同一组列上存在一个索引时，该索引就成为唯一索引且被该约束使用。

可以提取任何一个唯一约束，并将它用作主键。主键可以用作引用约束（以及其他唯一约束）中的父键。可在 CREATE TABLE 或 ALTER TABLE 语句中使用 PRIMARY KEY 子句来定义主键。主键可以由多个列组成。

主索引强制该主键的值为唯一的。当使用主键创建表时，数据库管理器会在该键上创建一个主索引。

用作唯一约束的索引的某些性能提示包括:

当初次装入带索引的空表时，LOAD 将提供比 IMPORT 更好的性能。不管是使用 LOAD 的 INSERT方式还是 REPLACE 方式，这种情况都一样。当把大量数据追加到一个带索引的现有表中（使用 IMPORT INSERT 或 LOAD INSERT）时，LOAD 的性能只比 IMPORT 的稍好一点。如果要使用 IMPORT 命令来进行大量数据的初始装入，那么要在导入或装入数据之后创建唯一键。这样避免了当装入该表时维护索引的额外开销。它还使索引使用最少量的存储器。如果正在以 REPLACE 方式使用 LOAD 实用程序，那么在装入数据之前创建唯一键。在这种情况下，在装入期间创建索引比在装入之后使用 CREATE INDEX 语句更有效。

## 设计主键约束

每个表都可以有一个主键。主键是与唯一约束具有相同属性的一个列或列的组合。可使用主键和外键约束来定义表之间的关系。

因为主键用来标识表中的一行，所以它应该是唯一的，并且只进行非常少的添加或删除。一个表不能有多个主键，但可以有多个唯一键。主键是可选的，可以在创建或改变表时使用 PRIMARY KEY 子句定义。当导出或重组数据时，主键可以对数据进行排序，所以它们也是有益的。

设计主键约束类似于设计唯一约束，如第 249 页的『设计唯一约束』中所述。唯一的差别在于每个表只能有一个主键约束，但可以有许多唯一约束。

**注:** 可以具有基于组合主键的主键约束。

## 设计检查约束

创建检查约束时，会出现下列两种情况之一: (i) 所有行都满足检查约束，或者 (ii) 部分或所有行不满足检查约束。

**所有行都满足检查约束**

当所有行都满足检查约束时，将成功创建检查约束。以后如果尝试插入或更新不满足约束业务规则的数据，那么这些尝试将被拒绝。

**某些行或所有行不满足检查约束**

当有部分行不满足检查约束时，将不会创建检查约束（也就是说，ALTER TABLE 语句将失败）。下面显示了用于将新约束添加至 EMPLOYEE 表的 ALTER TABLE 语句。该检查约束的名称为 CHECK_JOB。当 INSERT 或 UPDATE 语句失败时，数据库管理器将使用此名称来通知您违反了哪个约束。CHECK 子句用于定义表检查约束。

```
ALTER TABLE EMPLOYEE
 ADD CONSTRAINT check_job
 CHECK (JOB IN ('Engineer', 'Sales', 'Manager'));
```

因为已定义了表，所以使用 ALTER TABLE 语句。如果 EMPLOYEE 表中存在与要定义的约束相冲突的值，那么 ALTER STATEMENT 语句不会成功完成。

由于检查约束和其他类型的约束用于实施业务规则，所以可能需要常常对它们进行更改。当您所在组织中的业务规则更改时，就需要这样这样做。每次需要更改检查约束时，必须删除它，然后重新创建新的检查约束。随时都可以删除检查约束，并且此操作不会影响表或表中的数据。删除检查约束时，您必须了解该约束所执行的数据验证不再有效。

### 检查约束与前触发器的比较

在考虑使用触发器还是检查约束来保持数据完整性时，需要考虑检查约束之间的差别。

由于多个用户访问和更改数据，所以必须维护关系数据库中的数据完整性。每当共享数据时，就需要确保数据库中值的准确性。

**检查约束**

（表）检查约束对添加至特定表的数据设置限制。可以使用表检查约束来对表列中允许的值定义除数据类型限制之外的限制。表检查约束对同一表的同一行中的其他值执行范围检查或检查。

如果规则适用于使用数据的所有应用程序，那么可使用表检查约束来对表中允许的数据强制执行限制。表检查约束使得限制通常适用且更易于维护。

强制执行检查约束对于维护数据完整性来说很重要，但每次修改大量数据时，它也会产生一定的开销，这可能会影响性能。

**前触发器**

通过使用在更新或插入操作之前运行的触发器，可以在实际修改数据库之*前*修改要更新或插入的值。这些触发器可用来在需要时将应用程序中的输入（数据的用户视图）变换为内部数据库格式。前触发器还可用来使其他非数据库操作通过用户定义的函数被激活。

除了进行修改以外，前触发器的常见用法是使用 SIGNAL 子句进行数据验证。

当用于数据验证时，前触发器与检查约束之间存在以下两项差别:

1. 前触发器并不像检查约束那样仅限于访问同一个表的同一行中的其他值。
2. 执行 LOAD 操作之后，在对表执行 SET INTEGRITY 操作期间不会执行触发器（包括前触发器在内）。但是会验证检查约束。

## 设计外键（引用）约束

*引用完整性*是通过将外键（或引用）约束添加至表定义和列定义并对所有外键列创建索引而强加的。一旦定义了索引和外键约束，就会针对定义的约束检查对表和列中数据的更改。请求操作的完成取决于约束检查的结果。

引用约束是使用 CREATE TABLE 或 ALTER TABLE 语句中的 FOREIGN KEY 子句和 REFERENCES 子句建立的。创建引用约束之前，应考虑引用约束对类型表以及对作为类型表的父表的影响。

外键的标识在一个表的行内或两个表的行之间的值上施加约束。数据库管理器检查表定义中指定的约束，并相应地维持关系。目标是在不降低性能的条件下，无论何时一个数据库对象引用另一个数据库对象都要维持完整性。

例如，主键和外键各有一个部门号列。对于 EMPLOYEE 表，该列名为 WORKDEPT，而对于 DEPARTMENT 表，该列名为 DEPTNO。这两个表之间的关系由下列约束定义:

- 对于 EMPLOYEE 表中的每个职员只有一个部门号，且该编号存在于 DEPARTMENT表中。
- EMPLOYEE 表中的每一行都只与 DEPARTMENT 表中的一行相关。这两个表之间存在唯一的关系。
- 在 EMPLOYEE 表中具有 WORKDEPT 的非空值的每一行只与 DEPARTMENT 表的 DEPTNO 列中的一行相关。
- DEPARTMENT 表是父表，而 EMPLOYEE 表是从属表。

定义父表 DEPARTMENT 的语句如下所示:

```
CREATE TABLE DEPARTMENT
            (DEPTNO    CHAR(3)     NOT NULL,
              DEPTNAME  VARCHAR(29) NOT NULL,
              MGRNO     CHAR(6),
              ADMRDEPT  CHAR(3)     NOT NULL,
              LOCATION  CHAR(16),
                    PRIMARY KEY (DEPTNO))
      IN RESOURCE
```

定义从属表 EMPLOYEE 的语句如下所示:

```
CREATE TABLE EMPLOYEE
            (EMPNO     CHAR(     NOT NULL PRIMARY KEY,
              FIRSTNME   VARCHAR(12)  NOT NULL,
              LASTNAME   VARCHAR(15)  NOT NULL,
              WORKDEPT   CHAR(3),
              PHONENO    CHAR(4),
              PHOTO     BLOB(10m)   NOT NULL,
                    FOREIGN KEY DEPT (WORKDEPT)
                    REFERENCES DEPARTMENT ON DELETE NO ACTION)
      IN RESOURCE
```

通过将 DEPTNO 列指定为 DEPARTMENT 表的主键，而将 WORKDEPT 指定为 EMPLOYEE 表的外键，就对 WORKDEPT 值定义了引用约束。此约束实现这两个表的值之间的引用完整性。在这种情况下，添加至 EMPLOYEE 表的任何职员必须具有一个可以在 DEPARTMENT 表中找到的部门号。

职员表中的引用约束的删除规则为 NO ACTION，这表示如果一个部门中有任何职员，那么不能将该部门从 DEPARTMENT 表中删除。

虽然先前的示例使用 CREATE TABLE 语句来添加引用约束，但是也可以使用 ALTER TABLE 语句。

另一个示例: 使用与先前示例所用的相同表定义。另外，在 EMPLOYEE 表之前创建 DEPARTMENT 表。每个部门有一个经理，且该经理在 EMPLOYEE 表中列出。DEPARTMENT 表的 MGRNO 实际是 EMPLOYEE 表的外键。因为此引用循环，此约束存在一个小小的问题。可在以后添加外键。还可以使用 CREATE SCHEMA 语句来同时创建 EMPLOYEE 和 DEPARTMENT 表。

另请参阅第 255 页的『引用约束中的外键』。

## 触发器与引用约束之间的交互的示例

更新操作可能会导致触发器与引用约束和检查约束交互。

图 23 和相关描述是对更新数据库中的数据的语句所执行的具有代表性的处理。

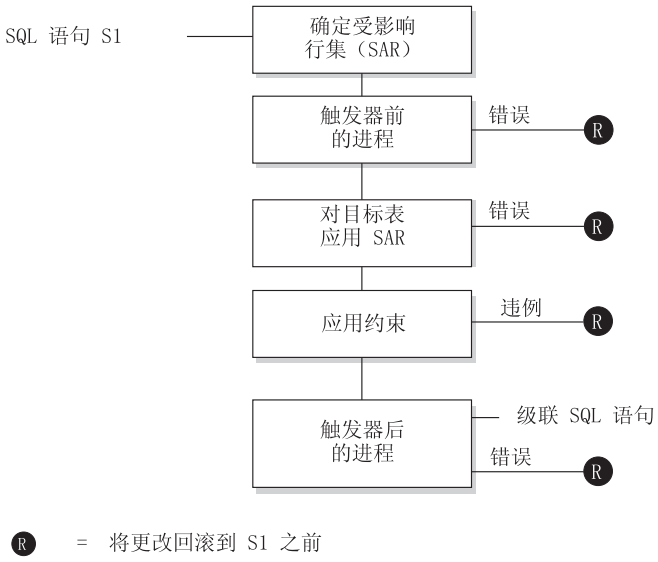



*图 23. 处理包含相关触发器和约束的语句*

图 23 显示用于处理更新表的语句的一般顺序。假定表包含级联的前触发器、引用约束、检查约束和后触发器。以下是对图 23 中的框和其他项的描述。

- 语句 $S_1$

  这是开始过程的 DELETE、INSERT 或 UPDATE 语句。在此描述中，语句 $S_1$ 标识一个称为*主题表*的表（或基于某个表的可更新视图）。

- 确定受影响行集

  此步骤是对 CASCADE 和 SET NULL 的引用约束删除规则以及对后触发器中的级联语句重复的过程的起始点。

  此步骤的用途是确定语句的*受影响行集*。包括的行集基于语句:

  – 对于 DELETE，受影响行集是符合语句的搜索条件的所有行（或定位 DELETE 的当前行）
  – 对于 INSERT，受影响行集是由 VALUES 子句或全查询标识的行
  – 对于 UPDATE，受影响行集是符合搜索条件的所有行（或定位 UPDATE 的当前行）。

如果受影响行集为空，那么将没有前触发器、没有适用于主题表的更改或没有要对语句处理的约束。

- 处理前触发器

  按创建日期的升序顺序处理所有前触发器。每个前触发器将对受影响行集中的每行处理一次触发操作。

  在处理触发操作期间可能会发生错误，在这种情况下，由于原始语句 $S_1$ 产生的所有更改（到目前为止的情况）都将回滚。

  如果没有前触发器或者受影响行集为空，那么将跳过此步骤。

- 将受影响行集应用于主题表

  使用受影响行集将实际删除、插入或更新操作应用于数据库中的主题表。

  应用受影响行集时可能会发生错误（例如，在唯一索引存在的情况下尝试插入具有重复键的行），在这种情况，由于原始语句 $S_1$ 产生的所有更改（到目前为止的情况）都将回滚。

- 应用约束

  如果受影响行集不为空，那么将应用与主题表关联的约束。这包括唯一约束、唯一索引、引用约束、检查约束和与视图上的 WITH CHECK OPTION 相关的检查。带有 CASCADE 或 SET NULL 删除规则的引用约束可能导致激活其他触发器。

  违反任何约束或 WITH CHECK OPTION 将产生错误，并且由于 $S_1$ 产生的所有更改（到目前为止的情况）都将回滚。

- 处理后触发器

  按创建日期的升序顺序处理 $S_1$ 激活的所有后触发器。

  即使受影响行集为空，FOR EACH STATEMENT 触发器也正好处理一次触发操作。FOR EACH ROW 触发器将对受影响行集中的每行处理一次触发操作。

  在处理触发操作期间可能会发生错误，在这种情况下，由于原始语句 $S_1$ 产生的所有更改（到目前为止的情况）都将回滚。

  触发器的触发操作可能包括一些触发语句，例如，DELETE、INSERT 或 UPDATE 语句。在此描述中，将每个这种语句都视为*级联语句*。

  级联语句是在后触发器的触发操作中处理的 DELETE、INSERT 或 UPDATE 语句。此语句开始级联级触发器处理。可以将此过程视为将触发语句指定为新的 $S_1$，并循环执行此处描述的所有步骤。

  处理完每个 $S_1$ 激活的所有后触发器中的所有触发语句后，对原始 $S_1$ 的处理就完成了。

- R = 将更改回滚到 $S_1$ 之前

  处理期间发生的任何错误（包括约束违例）将导致回滚由于原始语句 $S_1$ 直接或间接产生的所有更改。因此，数据库将返回到正好在执行原始语句 $S_1$ 之前所处的那个状态。

## 引用约束中的外键

外键在同一个或另一个表中引用主键或唯一键。外键的指定指示将根据指定的引用约束来维持该引用完整性。

可在 CREATE TABLE 或 ALTER TABLE 语句中使用 FOREIGN KEY 子句来定义外键。外键使它的表依赖于另一个称为父表的表。在一个表中，组成外键的列或一组列的值必须与父表的唯一键值或主键值匹配。

外键中的列数必须等于父表的对应主约束或唯一约束（称为父键）中的列数。另外，键列定义的对应部分必须具有相同的数据类型和长度。可以赋予外键一个*约束名*。如果未赋予名称，那么会自动赋予一个。为便于使用，建议赋予一个*约束名*，而不要使用系统生成的名称。

**如果**一个组合的外键每一列的值等于父键对应列的值，那么该外键的值与该父键的值匹配。包含空值的外键不能与父键的值匹配，因为定义的父键不能有空值。但是，一个空的外键值始终是有效的，无论它的任何一个非空部分的值如何。

下列规则适用于外键定义:

- 一个表可以有多个外键
- 如果任何部分都可空，那么该外键可空
- 如果任何部分为空，那么该外键值为空

在处理外键时，您可以执行下列操作:

- 创建带有或不带外键的表。
- 在创建或改变表时定义外键。
- 在改变表时删除外键。

## 实用程序操作的表约束隐含意义

如果正在装入行的表有引用完整性约束，那么 LOAD 实用程序就会使该表处于设置完整性暂挂状态以通知您需要对该表运行 SET INTEGRITY 语句，以便验证装入的行的引用完整性。LOAD 实用程序完成后，您需要发出 SET INTEGRITY 语句，以便对装入的行执行引用完整性检查以及使该表脱离设置完整性暂挂状态。

例如，如果 DEPARTMENT 和 EMPLOYEE 表是唯一处于设置完整性暂挂状态的表，那么可执行以下语句:

```
SET INTEGRITY FOR DEPARTMENT ALLOW WRITE ACCESS,
   EMPLOYEE ALLOW WRITE ACCESS,
   IMMEDIATE CHECKED FOR EXCEPTION IN DEPARTMENT,
   USE DEPARTMENT_EX,
   IN EMPLOYEE USE EMPLOYEE_EX
```

引用约束以下列方式影响 IMPORT 实用程序:

- 如果该对象表有该表以外的其他对象从属于它，那么不允许 REPLACE 和 REPLACE CREATE 函数。

  要使用这些函数，首先要删除该表为父表的所有外键。当导入完成时，使用 ALTER TABLE 语句重新创建这些外键。
- 导入带自引用约束的表的成功与否取决于这些行的导入顺序。

### 更改对象时的语句依赖性

语句依赖性包括程序包和高速缓存的动态 SQL 和 XQuery 语句。*程序包*是一个数据库对象，它包含数据库管理器以适合于特定应用程序的最有效方式访问数据所需的信息。*绑定*是创建程序包的过程，在执行应用程序时，数据库管理器需要这个程序包才能访问数据库。

程序包和高速缓存的动态 SQL 和 XQuery 语句可以依赖于许多类型的对象。

可以显式引用这些对象，例如，在一个 SQL SELECT 语句中涉及的一个表或用户定义的函数。也可以隐式引用这些对象，例如，当删除父表中的一行时，为确保不违反**引用约束**而需要检查的从属表。程序包还与已授予程序包创建者的特权有关。

如果程序包或高速缓存的动态查询语句依赖于某个对象而该对象被删除，那么该程序包或高速缓存的动态查询语句将被置于“无效”状态。如果程序包依赖于用户定义的函数而该函数被删除，那么在下列情况下，该程序包被置于“不可用”状态:

- 处于无效状态的高速缓存的动态 SQL 或 XQuery 语句在下次使用时将被自动重新优化。如果该语句需要的一个对象已被删除，那么执行该动态 SQL 或 XQuery 语句可能失败，并伴有错误消息。
- 处于无效状态的程序包在下次使用时将被隐式重新绑定。也可以显式重新绑定这种程序包。如果由于删除一个触发器而将一个程序包标记为无效，那么重新绑定后的程序包不再调用该触发器。
- 必须显式重新绑定处于不可用状态的程序包，然后才能使用。

联合数据库对象具有类似的依赖性。例如，删除服务器或者改变服务器定义将使引用与该服务器相关联的昵称的任何程序包或高速缓存动态 SQL 无效。

在某些情况下，重新绑定程序包是不可能的。例如，如果一个表已删除但尚未重新创建，那么不能重新绑定该程序包。在这种情况中，需要重新创建该对象或更改该应用程序，以使它不使用删除的对象。

在许多其他情况下，例如，如果删除了一个**约束**，那么重新绑定该程序包是可能的。

下列系统目录视图可帮助您确定程序包的状态和程序包的依赖性:

- SYSCAT.PACKAGEAUTH
- SYSCAT.PACKAGEDEP
- SYSCAT.PACKAGES

## 设计参考约束

在插入或更新记录时由数据库管理器强制执行的约束可能会产生大量系统开销，尤其在装入大量具有引用完整性约束的记录时。如果在将记录插入到表中之前应用程序已验证信息，那么使用参考约束比使用一般约束的效率要高。

参考约束告知数据库管理器数据遵从哪些规则，但数据库管理器不强制执行这些规则。但是，此信息可由 DB2 优化器使用并且可能产生较好的 SQL 查询性能。

以下示例说明参考约束的用途以及它们的工作方式。这个简单的表包含关于申请人的年龄和性别的信息:

```
CREATE TABLE APPLICANTS
  (
   AP_NO INT NOT NULL,
     GENDER  CHAR(1) NOT NULL,
CONSTRAINT GENDEROK
   CHECK (GENDER IN ('M', 'F'))
   NOT ENFORCED
   ENABLE QUERY OPTIMIZATION,
      AGE  INT NOT NULL,
CONSTRAINT AGEOK
   CHECK (AGE BETWEEN 1 AND 80)
   NOT ENFORCED
   ENABLE QUERY OPTIMIZATION,
 );
```

此示例包含两个子句，它们更改列约束的行为。第一个选项是 NOT ENFORCED，它在插入或更新数据时指示数据库管理器不要强制检查此列。

第二个选项是 ENABLE QUERY OPTIMIZATION，在对此表运行 SELECT 语句时数据库管理器将使用此选项。指定此值后，数据库管理器将在优化 SQL 时使用约束中的信息。

如果表包含 NOT ENFORCED 选项，那么 INSERT 语句的行为可能会很奇怪。在对 APPLICANTS 表运行以下 SQL 语句时，该语句不会导致任何错误:

```
INSERT INTO APPLICANTS VALUES
 (1, 'M', 54),
 (2, 'F', 38),
 (3, 'M', 21),
 (4, 'F', 89),
 (5, 'C', 10),
 (6, 'S',100),
```

申请人编号 5 的性别为 (C)，它表示小孩，而申请人编号 6 不仅具有不寻常的性别并且超过 AGE 列的年龄限制。在这两种情况下，数据库管理器允许进行插入操作，因为约束是 NOT ENFORCED。对该表执行 SELECT 语句的结果如下所示:

```
SELECT * FROM APPLICANTS
  WHERE GENDER = 'C';

APPLICANT  GENDER  AGE
---------  ------  ---

选择了 0 个记录。
```

数据库管理器对该查询返回不正确的答案，即使在表中找到值 'C' 亦如此，但此列上的约束告知数据库管理器唯一有效的值为 'M' 或 'F'。在优化语句时，ENABLE QUERY OPTIMIZATION 关键字也允许数据库管理器使用此约束信息。如果这不是您想要的行为，那么需要通过使用 ALTER TABLE 语句来更改约束，如下所示:

```
ALTER TABLE APPLICANTS
  ALTER CHECK AGEOK DISABLE QUERY OPTIMIZATION
```

如果重新发出该查询，那么数据库管理器将返回下列正确结果:

```
SELECT * FROM APPLICANTS
  WHERE SEC = 'C';

APPLICANT GENDER  AGE
---------  ------  ---
        5  C       10

1 record(s) selected.
```

如果您可以保证应用程序是插入和更新数据的唯一应用程序，那么此时是使用参考约束的最佳情形。如果应用程序事先已检查所有信息（如性别和年龄），那么使用参考约束可能会导致性能更高并且不会做重复工作。另一种可能使用参考约束的情形是设计数据仓库时。

## 创建和修改约束

可以使用 ALTER TABLE 语句将约束添加至现有表。

约束名不能与在 ALTER TABLE 语句内指定的任何其他约束相同，且必须在该表内是唯一的（这包括定义的任何引用完整性约束的名称）。在成功执行该语句之前，会对照新条件检查现有数据。

### 创建和修改唯一约束

可以将唯一约束添加至现有表。约束名不能与在 ALTER TABLE 语句内指定的任何其他约束相同，且必须在该表内是唯一的（这包括定义的任何引用完整性约束的名称）。在成功执行该语句之前，会对照新条件检查现有数据。

要使用命令行来定义唯一约束，可使用 ALTER TABLE 语句的 ADD CONSTRAINT 选项。例如，以下语句将一个唯一约束添加至 EMPLOYEE 表，它表示唯一标识该表中的职员的一个新方法:

```
ALTER TABLE EMPLOYEE
  ADD CONSTRAINT NEWID UNIQUE(EMPNO,HIREDATE)
```

要修改此约束，必须先删除此约束，然后重新创建。

### 创建和修改主键约束

可以将主键约束添加至现有表。约束名必须在表内是唯一的（这包括定义的任何引用完整性约束的名称）。在成功执行该语句之前，会对照新条件检查现有数据。

要使用命令行添加主键，请输入:

```
ALTER TABLE <name>
      ADD CONSTRAINT <column_name>
PRIMARY KEY <column_name>
```

不能修改现有约束。要将另一列或另一组列定义为主键，必须先删除现有主键定义，然后重新创建。

### 创建和修改检查约束

当添加表检查约束时，插入或更新该表的程序包和高速缓存的动态 SQL 可能被标记为无效。

要使用命令行来添加表检查约束，请输入:

```
ALTER TABLE EMPLOYEE
  ADD CONSTRAINT REVENUE CHECK (SALARY + COMM > 25000)
```

要修改此约束，必须先删除此约束，然后重新创建。

### 创建和修改外键（引用）约束

外键是对另一个表中的数据值的引用。有不同类型的外键约束。

当将一个外键添加至表时，包含下列语句的程序包和高速缓存的动态 SQL 可能被标记为无效:

- 插入或更新包含外键的表的语句

- 更新或删除父表的语句。

要使用命令行添加外键，请输入:

```
ALTER TABLE <name>
ADD CONSTRAINT <column_name>
 FOREIGN KEY <column_name>
 ON DELETE <action_type>
 ON UPDATE <action_type>
```

以下示例显示 ALTER TABLE 语句如何将主键和外键添加至一个表:

```
 ALTER TABLE PROJECT
ADD CONSTRAINT PROJECT_KEY
    PRIMARY KEY (PROJNO)
ALTER TABLE EMP_ACT
  ADD CONSTRAINT ACTIVITY_KEY
      PRIMARY KEY (EMPNO, PROJNO, ACTNO)
  ADD CONSTRAINT ACT_EMP_REF
      FOREIGN KEY (EMPNO)
      REFERENCES EMPLOYEE
      ON DELETE  RESTRICT
  ADD CONSTRAINT ACT_PROJ_REF
      FOREIGN KEY (PROJNO)
      REFERENCES PROJECT
      ON DELETE CASCADE
```

要修改此约束，必须先删除此约束，然后重新创建。

**创建和修改参考约束**

为了提高查询性能，可以向表添加参考约束。当对 DDL 指定 NOT ENFORCED 选项时，可以使用 CREATE TABLE 或 ALTER TABLE 语句添加参考约束。

**限制**: 对表定义参考约束后，只有先除去参考约束才能改变该表的列名。

要使用命令行对表指定参考约束，请对一个新表输入以下命令:

```
ALTER TABLE <name> <constraint attributes> NOT ENFORCED
```

ENFORCED 或 NOT ENFORCED: 指定数据库管理器在正常操作（如插入、更新或删除）期间是否强制执行约束。

- 不能对函数依赖性指定 ENFORCED（SQLSTATE 42621）。
- 仅当已知表数据以独立的方式符合约束时，才应指定 NOT ENFORCED。如果数据实际上不符合该约束，那么查询结果可能是不可预测的。

要修改此约束，必须先删除此约束，然后重新创建。

## 查看表的约束定义

可以在 SYSCAT.INDEXES 和 SYSCAT.REFERENCES 目录视图中找到表的约束定义。

SYSCAT.INDEXES 视图的 UNIQUERULE 列指示索引的特征。如果此列的值为 P，那么索引为主键；如果该值为 U，那么索引是唯一索引（但不是主键）。

SYSCAT.REFERENCES 目录视图中包含引用完整性（外键）约束信息。

## 删除约束

可以使用 ALTER TABLE 语句显式删除表检查约束，或作为 DROP TABLE 语句的结果将其隐式删除。

要删除约束，使用带有 DROP 或 DROP CONSTRAINT 子句的 ALTER TABLE 语句。这将允许您“绑定”并继续访问包含受影响列的表。一个表上的所有唯一约束的名称可以在 SYSCAT.INDEXES 系统目录视图中找到。

### 删除唯一约束

可以使用 ALTER TABLE 语句来显式删除唯一约束。

ALTER TABLE 语句的 DROP UNIQUE 子句删除唯一约束 constraint-name 以及依赖于此唯一约束的所有引用约束的定义。constraint-name 必须标识现有唯一约束。

```
ALTER TABLE <table-name>
  DROP UNIQUE <constraint-name>
```

删除此唯一约束使使用该约束的任何程序包或高速缓存的动态 SQL 无效。

### 删除主键约束

使用 ALTER TABLE 语句的 DROP PRIMARY KEY 子句删除主键约束。

ALTER TABLE 语句的 DROP PRIMARY KEY 子句删除主键以及依赖于此主键的所有引用约束的定义。表必须具有主键。要使用命令行来删除主键，请输入:

```
ALTER TABLE <table-name>
       DROP PRIMARY KEY
```

### 删除（表）检查约束

删除检查约束时，与该表有 INSERT 或 UPDATE 关系的所有程序包和高速缓存的动态语句将变得无效。一个表上的所有检查约束的名称可以在 SYSCAT.INDEXES 目录视图中找到。在尝试删除带有系统生成的名称的表检查约束之前，在 SYSCAT.CHECKS 目录视图中查找该名称。

以下语句将删除检查约束 constraint-name。constraint-name 必须标识在表上定义的现有检查约束。要使用命令行来删除表检查约束:

```
 ALTER TABLE <table_name>
DROP CHECK <check_constraint_name>
```

或者，可以使用带有 DROP CONSTRAINT 选项的 ALTER TABLE 语句。

### 删除外键（引用）约束

使用 ALTER TABLE 语句的 DROP CONSTRAINT 子句来删除外键约束。

ALTER TABLE 语句的 DROP CONSTRAINT 子句删除约束 *constraint-name*。*constraint-name* 必须标识对表定义的现有外键约束、主键或唯一约束。要使用命令行来删除外键，请输入:

```
ALTER TABLE <table-name>
  DROP FOREIGN KEY <foreign_key_name>
```

下列示例在 ALTER TABLE 语句中使用 DROP PRIMARY KEY 和 DROP FOREIGN KEY 子句来删除表上的主键和外键:

```
ALTER TABLE EMP_ACT
    DROP PRIMARY KEY
    DROP FOREIGN KEY ACT_EMP_REF
    DROP FOREIGN KEY ACT_PROJ_REF
ALTER TABLE PROJECT
    DROP PRIMARY KEY
```

当删除外键约束时，包含下列语句的程序包或高速缓存的动态语句可能被标记为无效:

- 插入或更新包含外键的表的语句
- 更新或删除父表的语句。

# 第 13 章 索引

*索引*是一个或多个键的集合，每个键指向表中的一行。SQL *优化器*自动选择最有效率的访问表中数据的方法。当确定最快速的数据访问路径时，优化器会将索引考虑在内。

**注:** 并不是所有索引都指向表中的行。MDC 块索引就指向数据的扩展数据块（或块）。XML 数据的 XML 索引使用特定的 XML 模式表达式来为存储在单个列中的 XML 文档中的路径和值建立索引。该列的数据类型必须是 XML。MDC 块索引和 XML 索引都是由系统生成的索引。

索引由数据库管理器用来:

- 提高性能。在大多数情况下，使用索引访问数据的速度更快。虽然不能为视图创建索引，但为视图所基于的表创建的索引有时可以提高该视图的操作性能。
- 确保唯一性。具有唯一索引的表不能包含具有完全相同的键的行。

将数据添加到表时，除非已对该表和正在添加的数据执行了其他操作，否则简单地将数据追加至该表的底部。数据是无序的。搜索特定数据行时，必须检查从第一行到最后一行的每个表行。将索引用作按顺序访问表中的数据的一种方法，该顺序在其他情况下可能不可用。

可以使用数据行中的列值来标识整个行。可能需要一列或多列来标识该行。这种列称为*键*。一个列可以在多个键中使用。

索引按键中的值进行排序。键可以是唯一的，也可以是非唯一的。每个表应该至少有一个唯一键；但还可以有其他非唯一键。每个索引正好有一个键。例如，可以使用职员标识编号（唯一）作为一个索引的键，并使用部门号（非唯一）作为另一个索引的键。

## 示例

在第 264 页的图 24 中，表 A 的一个索引基于表中的职员编号。此键值提供指向表行的指针。例如，职员编号 19 指向职员 KMP。索引允许通过指针创建指向数据的路径有效地访问表中的各行。

可创建唯一索引以确保索引键的唯一性。*索引键*是定义了索引的一个列或一些列的有序集合。使用唯一索引将确保在编入索引的列中，每个索引键的值都是唯一的。

第 264 页的图 24 显示了索引与表之间的关系。

© Copyright IBM Corp. 1993, 2010

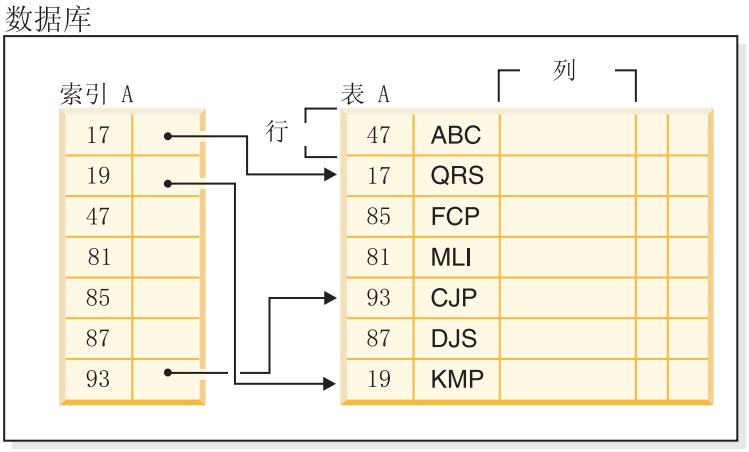



*图 24. 索引与表之间的关系*

图 25 说明一些数据库对象之间的关系。它还显示了表、索引和长型数据存储在表空间中的情况。

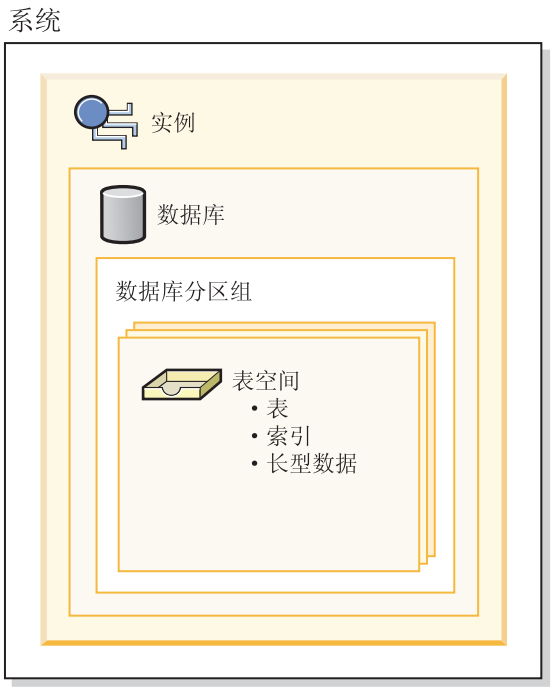



*图 25. 所选择的数据库对象之间的关系*

# 索引的类型

有五种类型的索引：唯一索引、非唯一索引、集群索引、非集群索引以及系统为多维集群（MDC）表生成的块索引。

## 唯一索引和非唯一索引

唯一索引是这样一种索引，它通过确保表中没有两个数据行具有完全相同的键值来帮助维护数据完整性。

尝试为已经包含数据的表创建唯一索引时，将检查组成该索引的列中的值是否唯一；如果该表包含具有重复键值的行，那么索引创建过程将失败。为表定义了唯一索引之后，每当在索引中添加或更改键时就会强制唯一性。（这包括插入、更新、装入、导入和设置完整性以命名一部分。）除了强制数据值的唯一性以外，唯一索引还可用来提高查询处理期间检索数据的性能。

另一方面，非唯一索引不用于对与它们关联的表强制执行约束。相反，非唯一索引通过维护频繁使用的数据值的排序顺序，仅仅用于提高查询性能。

## 集群索引和非集群索引

索引体系结构分为集群或非集群。集群索引是这样的索引：数据页中的行的顺序对应于索引中的行的顺序。这就是为何给定表中只能存在一个集群索引，而表中可以存在多个非集群索引。在某些关系数据库管理系统中，集群索引的叶子节点对应于实际数据，而不是对应于指向位于其他地方的数据的指针。

集群索引和非集群索引都只包含索引结构中的键和记录标识。记录标识始终指向数据页中的行。集群索引和非集群索引的位于区别在于：数据库管理器尝试按照相应的键在索引页中的出现顺序来将数据保存在数据页中。因此，数据库管理器将尝试把具有相似键的行插入同一页中。如果对表进行了重组，那么会按照索引键的顺序将行插入数据页中。

重组与所选索引相关的表将对数据重新建立集群。建立了集群的索引对于带有范围谓词的列非常有用，因为它允许对表中的数据作更有效的顺序访问。由于相似的值在同一数据页上，从而减少页访存次数。

通常，表中只有一个索引可以具有较高的集群度。

## 使用集群索引提高性能

由于集群索引使存储在页面中的数据的访问路径更线性化，所以它们可以提高大多数查询操作的性能。此外，由于具有相似索引键值的行都存储在一起，所以在使用集群索引时预取效率通常更高。

但是，不能将集群索引指定为与 CREATE TABLE 语句配合使用的表定义的一部分。相反，只通过执行指定了 CLUSTER 选项的 CREATE INDEX 语句来创建集群索引。然后，ALTER TABLE 语句应用于在表中添加与创建的集群索引对应的主键。然后，此集群索引将用作表的主键索引。

**注**：通过使用 ALTER TABLE 语句将表中的 PCTFREE 设置为适当的值，并保留足够的可用空间以将具有相似值的行插入页中，有助于使表保持分群。有关更多信息，请参阅 ALTER TABLE 语句以及《*调整数据库性能*》中的『减少重组表和索引的需要』。

通常，如果集群索引是唯一的，那么集群维护起来就更有效率。

### 主键或唯一键约束与唯一索引之间的差别

了解主唯一键约束与唯一索引之间没有很大差别这一点很重要。数据库管理器使用唯一索引和 NOT NULL 约束的组合来实现主键约束和唯一键约束的关系数据库概念。因此，唯一索引本身不强制执行主键约束，因为它们允许空值。（虽然空值表示未知值，但在建立索引时，将一个空值视为与其他空值相同。）

因此，如果唯一索引由单个列组成，那么只允许一个空值 - 多个空值将违反唯一约束。同样，如果唯一索引由多个列组成，那么值和空值的特定组合只能使用一次。

### 双向索引

缺省情况下，双向索引允许按正反两个方向进行扫描。CREATE INDEX 语句的 ALLOW REVERSE SCANS 子句同时启用正向和反向索引扫描，也就是说，按创建索引时的顺序和相反（或反向）顺序。此选项允许您:

- 便于使用 MIN 和 MAX 函数
- 访存先前的键
- 不需要数据库管理器创建临时表来进行反向扫描
- 消除冗余反向顺序索引

如果指定了 DISALLOW REVERSE SCANS，那么不能反向扫描索引。（但实际上它将与 ALLOW REVERSE SCANS 索引完全相同。）

## 设计索引

索引通常用于加速对表的访问。但是，逻辑数据设计也可以使用索引。

例如，唯一索引不允许列中存在重复值的条目，从而保证了一个表中不会有两行相同。还可以创建索引，以将一列中的值按升序或降序进行排序。

**注**: 在创建索引时要记住，虽然它们可以提高读取性能，但会对写性能产生负面影响。这是因为对于数据库管理器写入表中的每行，它还必须更新任何受影响的索引。因此，只有在会明显提高整体性能时，才应创建索引。

在创建索引时，还必须考虑表结构和最常对这些表执行的查询的类型。例如，经常发出的查询的 WHERE 子句中出现的列是索引的优秀候选者。但是，在较少运行的查询中，索引对 INSERT 和 UPDATE 语句的性能产生的负面影响可能超过所带来的好处。

同样，在经常运行的查询的 GROUP BY 子句中出现的列可能会从创建索引中获益，尤其在用于分组行的值的数目小于要分组的行数时。

### 设计索引时的准则和注意事项

- *索引*由表中的列定义。它可以由表的创建者或知道某些列需要直接访问的用户来定义。除非已经存在用户定义的索引，否则会根据主键来自动创建主索引键。
- *索引键*是定义了索引的一个列或一些列的集合，它决定索引的有用程度。虽然构成一个索引键的列的顺序不会影响索引键的创建，但是当它决定是否使用索引时就可能影响优化器。

- 可在特定表上定义任意个索引，且这些索引能提高查询性能。索引管理器必须在更新、删除和插入操作期间维护索引。为接收很多更新内容的表创建大量索引可能减慢请求的处理速度。同样，大型索引键也会减慢处理请求的速度。因此，仅当频繁访问有明显有利之处时，才使用索引。
- 不是唯一索引键的一部分但要在该索引中存储或维护的列数据称为包含列。只能为唯一索引指定包含列。当用包含列创建索引时，仅对唯一键列进行排序并考虑其唯一性。使用包含列可以启用仅访问索引来进行数据检索，从而提高性能。
- 如果要建立索引的表是空的，那么仍会创建索引，但是在装入该表或插入行之前，不会建立任何索引条目。如果该表不为空，那么数据库管理器将在处理 CREATE INDEX 语句时创建索引条目。
- 对于*集群索引*，数据库管理器会尝试将表的新行插入到具有（由索引定义的）相似键值的现有行附近。
- 如果要让*主键索引*成为集群索引，那么不应在 CREATE TABLE 语句中指定主键。一旦创建了主键，就不能修改相关的索引。而是发出不带主键子句的 CREATE TABLE。然后，发出 CREATE INDEX 语句，并指定集群属性。最后，使用 ALTER TABLE 语句添加与刚创建的索引对应的主键。将把此索引用作主键索引。
- 索引会消耗磁盘空间。该磁盘空间大小取决于键列的长度和要建立索引的行数。随着插入到表中的数据增多，索引大小也会增加。因此，在规划数据库大小时，应考虑正在建立索引的数据量。下面是一些建立索引大小的注意事项:
  - 主键和唯一键约束始终创建系统生成的唯一索引。
  - 创建 MDC 表时也将创建由系统生成的块索引。
  - XML 列始终导致创建系统生成的索引。
  - 对外键约束列创建索引通常会有好处。

  **注**: 索引中的最大列数为 64。但是，如果对类型表建立索引，那么索引中的最大列数为 63。索引键的最大长度（包括所有开销）为 *indexpagesize*/4。表上允许的最大索引数为 32767。索引键的最大长度不能大于页大小的索引键长度限制。有关列存储长度的信息，请参阅“CREATE TABLE 语句”。有关键长度限制，请参阅『SQL 和 XQuery 限制』主题。

  **注**:

## 用于设计索引的工具

创建表后，需要考虑数据库管理器能够从这些表中检索数据的速度。可以使用“设计顾问程序”或 db2advis 命令来帮助您设计索引。

对表创建有用的索引可以极大地提高查询性能。与书籍的索引一样，使用表索引就可以通过最少的搜索而快速找到特定信息。使用索引从表中检索特定行可以减少数据库管理器需要执行的成本较高的输入/输出操作数。这是因为索引允许数据库管理器通过读取相对较少的数据页来找到行，而不是彻底搜索所有数据页直到找到所有匹配项为止。

DB2 设计顾问程序是一个工具，可帮助您显著提高工作负载性能。选择要为复杂工作负载创建哪些索引、MQT、集群维或数据库分区的任务可能会令人十分头痛。“设计顾问程序”标识了提高工作负载性能所需要的所有对象。如果工作负载中存在一组 SQL 语句，那么“设计顾问程序”将为下列各项提供一些建议:

- 新的索引

- 新的具体化查询表 (MQT)
- 对多维集群（MDC）表的转换
- 表的重新分发
- （通过 GUI 工具）删除指定的工作负载未使用的索引和 MQT

可以使用“设计顾问程序”立即实现这些建议中的一部分建议或所有建议，也可以安排稍后实现这些建议。

“设计顾问程序”可以使用“设计顾问程序”GUI 或命令行工具来帮助简化下列任务:

- 规划或建立新的数据库
- 工作负载性能调整

## 索引的空间需求

设计索引时，需要了解它们的空间需求。

对于每个索引，可以按如下公式估计所需的空间:

```
(平均索引键大小 + 索引键开销) * 行数 * 2
```

其中:

- “平均索引键大小”是索引键中每列的字节计数。（估计 VARCHAR 和 VARGRAPHIC 列的平均列大小时，使用当前数据大小的平均值加上两个字节。不要使用最大声明大小。）
- “索引键开销”取决于对其创建所有的表的类型。对于大型表（带或者不带 XML 索引），值为 11；除非表是分区表，这种情况下值为 13。对于常规表，如果不带 XML 索引，那么值为 9，如果带 XML 索引，那么值为 11。对于所有分区常规表，值为 11。
- 因子“2”表示开销，如非叶子页和可用空间。

**注:**

1. 对于允许空值的每个列，为空指示符添加一个额外的字节。
2. 对于在内部为多维集群（MDC）表创建的块索引，“行数”将被替换为“块数”。

对于 XML 列的每个索引，可以按如下公式估计所需的空间:

```
(平均索引键大小 + 索引键开销) * 建立索引的节点数 * 2
```

其中:

- “平均索引键大小”是组成索引的键部件的总和。XML 索引由多个 XML 键部件和一个值（sql-data-type）组成:

  ```
  固定开销 + 可变开销 + sql-data-type 的字节数
  ```

  其中:

  – “固定开销”是 14 个字节。
  – “可变开销”是建立索引的节点的平均深度加上 4 个字节。
  – sql-data-type 值的字节数与 SQL 遵循相同的规则。
- “建立索引的节点数”是要插入的文档数乘以样本文档中满足索引定义中的 XML 模式表达式（XMLPATTERN）的节点数所获得的结果。

在版本 8 之前创建的索引（1 类索引）与在版本 8（2 类索引）及其后续版本创建的索引不同。要了解一个表存在什么类型的索引，可使用 ADMIN_GET_TAB_INFO 表函数。要将 1 类索引转换为 2 类索引，可使用 REORG INDEXES .... CONVERT 命令。

在使用 REORG INDEXES 命令时，确保存储索引的表空间中有足够的可用空间。可用空间的大小应等于索引的当前大小。如果选择使用 ALLOW WRITE ACCESS 选项来重组索引，那么可能需要更多空间。这些空间用于存储重组索引期间影响索引的活动的日志。

创建索引时，临时空间是必需的。在创建索引期间所需的最大临时空间可以按如下公式估计:

```
(平均索引键大小 + 索引键开销) * 行数 * 3.2
```

或者

```
(平均索引键大小 + 索引键开销) * 建立索引的节点数 * 3.2
```

其中，因子“3.2”表示索引开销以及索引创建期间进行排序所需的空间。

**注**: 对于非唯一索引，存储重复键条目只需五个字节。上面显示的估计是假定没有重复的条目。因此以上公式可能会过多地估计存储索引所需的空间。

如果索引节点数超过 64 KB 数据，那么插入时将需要临时空间。可以按如下公式估计所需的临时空间:

```
(平均索引键大小) * 建立索引的节点数 * 1.2
```

可使用下面两个公式来估计每个叶子页的键数（第二个公式提供更准确的估计）。这些估计的准确度很大程度上取决于平均值反映实际数据的准确程度。

**注**: 对于 SMS 表空间，叶子页所需的最小空间为 12 KB。对于 DMS 表空间，最小值是一个扩展数据块。

- 每个叶子页的平均键数的粗略估计是:

$$\frac{(.9 * (U - (M*2))) * (D + 1)}{K + 7 + (5 * D)}$$

  其中:

  – U（一页上的可用空间）大约等于页大小减 100。对于 4096 的页大小，U 是 3996。
  – M = U / (9 + *最小键大小*)
  – D = 每个键值的平均重复项数目
  – K = *平均键大小*

  记住，*最小键大小*和*平均键大小*必须有一个额外字节，表示每个可空键部分；还必须有两个额外字节，表示每个变长键部分的长度。

  如果存在包含列，那么在计算*最小键大小*和*平均键大小*时应将它们考虑在内。

  “minimumKeySize”是组成索引的键部件的总和:

  ```
  固定开销 + 可变开销 + sql-data-type 的字节数
  ```

  其中:

– “固定开销”是 13 个字节。
– “可变开销”是建立索引的节点的最小深度加上 4 个字节。
– sql-data-type 值的字节数与 SQL 遵循相同的规则。

如果在创建索引期间指定了非缺省值 10% 的一个可用百分比值，那么可以使用任何 (100 - pctfree)/100 值替换 .9。

- 每个叶子页的平均键数的更准确估计是:

```
L = 叶子页数 = X / (叶子页上的平均键数)
```

其中，X 是表中的总行数。

对于 XML 列的索引，X 是该列中建立索引的节点总数。

可按如下方法估算索引的原始大小:

```
(L + 2L/(叶子页上的平均键数)) * 页大小
```

对于 DMS 表空间，将一个表上所有索引的大小加在一起，然后四舍五入为该索引所在表空间的扩展数据块大小的一个倍数。

应该为 INSERT/UPDATE 活动所引起的索引增长提供其他空间，这种增长可能导致分页。

使用以下计算方法来获得更精确的原始索引大小的估算值，以及该索引中级别数的估算值。（如果索引定义中使用包含列，可能要引起特别注意。）每个非叶子页的平均键数大约是:

```
   (.9 * (U - (M*2))) * (D + 1)
---------------------------
              K + 13 + (9 * D)
```

其中:

– U（一页上的可用空间）大约等于页大小减 100。对于 4096 的页大小，U 是 3996。
– D 是在非叶子页上每个键值的重复值的平均数目（这将比在叶子页上的小很多，您可能想将该值设置为 0 以便简化计算）。
– M = U / (9 + 非叶子页的*最小键大小*)
– K = 非叶子页的*平均键大小*

只要没有包含列，非叶子页与叶子页的*最小键大小*和*平均键大小*将是相同的。包含列不存储在非叶子页上。

除非 (100 - pctfree)/100 大于 .9，否则不应使用它来替换 .9，因为在创建索引期间会在非叶子页上留下最多 10% 的可用空间。

可用如下所示的方法估算非叶子页数:

```
if L > 1 then {P++; Z++}
While (Y > 1)
{
   P = P + Y
   Y = Y / N
  Z++
}
```

其中:

– P 是页数（最初为 0）。

– L 是叶子页数。

– N 是每个非叶子页的键数。

– Y = L / N

– Z 是索引树中的级别数（最初为 1）。

总页数是:

```
T = (L + P + 2) * 1.0002
```

附加的 0.02% 表示开销，包括空间映射页。

创建索引所需的空间容量估算为:

```
T * 页大小
```

## 创建索引

可以创建索引，以将一列中的值按升序或降序进行排序。可以使用 CREATE INDEX 语句、DB2 设计顾问程序或者 db2advis 设计顾问程序命令来创建索引。

例如，要从命令行中使用 CREATE INDEX 语句来创建索引，请输入:

```
CREATE UNIQUE INDEX EMP_IX
  ON EMPLOYEE(EMPNO)
  INCLUDE(FIRSTNAME, JOB)
```

INCLUDE 子句指定将附加列追加到一组索引键列，此子句仅适用于唯一索引。使用此子句所包含的任何列未用来强制唯一性。包含的这些列可能会通过仅访问索引而提高某些查询的性能。此选项可以:

- 不需要访问数据页来进行更多查询
- 消除冗余索引

如果对此索引所在的表发出了 SELECT EMPNO、FIRSTNAME 或 JOB FROM EMPLOYEE，那么可从该索引检索到所有必需的数据，而不需要读取数据页。这将提高性能。

**注**: 对于在版本 8 或更高版本中创建的索引（称为 2 类索引），删除或更新行时，只是将键标记为已删除。这称为伪删除键。在落实删除或更新操作后完成清除之前，不会从页中物理除去这些键。这种清除可由后续事务完成，该事务更改键在其中标记为已删除的页。可使用 REORG INDEXES 实用程序的 CLEANUP ONLY ALL 选项显式触发伪删除键的清除。

使用相同的 CREATE INDEX 语句来构建分区数据库环境中表的索引。索引中的数据根据表的分布键进行分布。完成此操作之后，将在数据库分区组中的每个数据库分区上创建一个 B+ 树。每个 B+ 树对属于该数据库分区的那部分表建立索引。在多分区数据库上定义的唯一索引中的列必须是分布键中列的超集。

**注**: 在版本 9.5 中，在 Solaris 平台上，如果使用了原始设备，那么 CREATE INDEX 语句将被挂起。Sun 公司将修正此问题并在内核补丁中发布该修订。

## 修改索引

如果要修改索引，那么必须先删除该索引，然后再次将它创建。没有 ALTER INDEX 语句。

例如，不能在未删除先前定义并创建新索引的情况下将列添加至键列的列表。可以使用 COMMENT 语句添加注释来描述索引的用途。

## 重命名索引

可以使用 RENAME 语句来重命名现有索引。

要重命名现有索引，请在命令行中发出以下语句:

```
RENAME INDEX <source index name> TO <target index name>
```

- <source index name> 是要重命名的现有索引的名称。该名称（包括模式名）必须标识数据库中已存在的索引。它不能是已声明的全局临时表的索引名。模式名不能是 SYSIBM、SYSCAT、SYSFUN 或 SYSSTAT。
- <target index name> 为索引指定不包含模式名的新名称。源对象的模式名用于限定对象的新名称。限定名不能标识数据库中已存在的索引。

重命名索引时，源索引不能是系统生成的索引。如果该语句成功，那么系统目录表将更新以反映新的索引名。

## 重建索引

因为索引不是在执行前滚操作期间创建的，所以某些数据库操作（例如，通过未完全记录的创建索引操作来进行前滚）可能会导致索引对象变得无效。可以通过在索引对象中重建索引来恢复此索引对象。

当数据库管理器检测到索引不再有效时，它会自动尝试重建索引。进行重建时，它受数据库或数据库管理器配置文件的 **indexrec** 参数控制。此参数有五个可能的设置:

- SYSTEM
- RESTART
- RESTART_NO_REDO
- ACCESS
- ACCESS_NO_REDO

RESTART_NO_REDO 和 ACCESS_NO_REDO 类似于 RESTART 和 ACCESS。

NO_REDO 选项意味着: 即使在执行原始操作（例如，CREATE INDEX）期间完全记录了索引，在前滚期间也不会重建索引，但是在重新启动或者首次访问时将创建索引。有关更多信息，请参阅 **indexrec** 参数。

如果数据库重新启动时间不重要，那么最好在数据库返回到一致状态的过程中重建无效索引。使用此方法时，重新启动数据库所需的时间将由于重新创建索引而较长；但一旦数据库返回到一致状态，正常处理将不受影响。

另一方面，如果在访问索引时重建它们，那么重新启动数据库所用的时间将较短，但响应时间可能会由于重新创建索引而意外变长；例如，访问具有无效索引的表的用户

必须等待重建索引。此外，在重新创建无效索引后，可能会获得意外锁定并且该锁定可能会挂起较长时间，尤其在导致重新创建索引的事务从不终止时（也就是说，落实或回滚所作的更改）。

## 删除索引

不能更改索引定义的任何子句；必须先删除然后再次创建该索引。（删除索引不会导致删除任何其他对象，但是可能会导致某些程序包无效。）使用 DROP 语句来删除索引。

不能显式删除主键或唯一键索引。必须使用下列其中一种方法删除它:

- 如果主索引或唯一约束是为主键或唯一键自动创建的，那么删除主键或唯一键将会使该索引也被删除。使用 ALTER TABLE 语句来执行删除。
- 如果主索引或唯一约束是用户定义的，那么必须使用 ALTER TABLE 语句先将主键或唯一键删除。在删除主键或唯一键之后，该索引就不再被认为是主索引或唯一索引，这时可以将其显式删除。

要使用命令行来删除索引，请输入:

```
DROP INDEX <index_name>
```

以下语句删除称为 PH 的索引:

```
DROP INDEX PH
```

任何从属于删除的索引的程序包和高速缓存的动态 SQL 和 XQuery 语句都被标记为无效。应用程序不受添加或删除索引所导致的更改的影响。

# 第 14 章 触发器

*触发器*定义一组操作，在响应对指定表的插入、更新或删除操作时将执行这些操作。执行这样的 SQL 操作时，触发器被认为是已*激活的*。触发器是可选的，并且可使用 CREATE TRIGGER 语句定义。

可将触发器与引用约束和检查约束配合使用，以强制执行数据完整性规则。还可使用触发器来导致更新其他表、自动生成或变换插入或更新的行的值或者调用函数以执行如发出警报之类的任务。

对于定义或强制*事务性*业务规则，触发器是非常有用的机制，这些规则涉及数据的不同状态（例如，薪水增长不能超过 10%）。

使用触发器会设置逻辑以在数据库内强制使用业务规则。这表示应用程序不负责强制使用这些规则。对所有表强制使用的集中逻辑意味着更容易维护，因为在逻辑更改时，不需要更改应用程序。

下列各项是在创建触发器时指定的:

- *主题表*指定对其定义触发器的表。
- *触发事件*定义修改主题表的特定 SQL 操作。该事件可以是插入、更新或删除操作。
- *触发器激活时间*指定触发器应在触发事件发生之前还是之后激活。

导致触发器激活的语句包括*一组受影响的行*。这些行就是正对其进行插入、更新或删除操作的主题表的行。*触发器粒度*指定触发器的操作是对该语句执行一次，还是对每个受影响的行执行一次。

*触发操作*包括可选搜索条件和每次激活触发器时执行的一组 SQL 语句。仅当搜索条件求值为 true 时，才执行这些 SQL 语句。如果触发器激活时间是在触发器事件之前，那么触发操作可包括用于进行选择、设置转换变量或发信号表明 SQL 状态的语句。如果触发器激活时间是在触发器事件之后，那么触发操作可包括用于进行选择、插入、更新、删除或发信号表明 SQL 状态的语句。

触发操作可使用*转换变量*来引用一组受影响的行中的值。转换变量使用由指定的名称限定的主题表中的各个列名，以标识是引用旧值（更新前）还是引用新值（更新后）。在之前、插入或更新触发器中，还可使用 SET Variable 语句来更改新值。

引用一组受影响的行中的各个值的另一种方法是使用*转换表*。转换表同样使用主题表中的各个列名，但它指定一个名称，以允许将一整组受影响的行视作一个表。只能在后触发器中使用转换表（也就是说，不能在前触发器和 INSTEAD OF 触发器中使用），并且可以对旧值和新值定义单独的转换表。

可以对表、事件（INSERT、UPDATE 和 DELETE）或激活时间（BEFORE、AFTER 和 INSTEAD OF）的组合指定多个触发器。当对特定表、事件和激活时间存在多个触发器时，激活触发器的顺序与创建它们的顺序相同。因此，最新创建的触发器就是最后激活的触发器。

激活触发器可能导致*触发器级联*，这是由于激活了一个执行 SQL 语句的触发器，这些 SQL 语句导致激活其他触发器或再次激活相同触发器。触发操作还可能导致对删除应用

© Copyright IBM Corp. 1993, 2010

引用完整性规则而产生的更新，从而可能导致激活更多触发器。借助触发器级联，可能会激活触发器和引用完整性删除规则链，从而由于单个 INSERT、UPDATE 或 DELETE 语句而导致对数据库的大幅度更改。

当多个触发器对同一对象执行插入、更新或删除操作时，临时表将用于避免访问冲突。这样可能会对性能产生显著影响，在分区数据库环境中尤其如此。

## 触发器的类型

*触发器*定义一组操作，在响应对指定表的插入、更新或删除操作时将执行这些操作。执行这样的 SQL 操作时，触发器被认为是已*激活的*。触发器是可选的，并且可使用 CREATE TRIGGER 语句定义。

可将触发器与引用约束和检查约束配合使用，以强制执行数据完整性规则。还可使用触发器来导致更新其他表、自动生成或变换插入或更新的行的值或者调用函数以执行如发出警报之类的任务。

支持下列类型的触发器:

**前触发器**

在更新或插入操作前运行。在实际修改数据库之前，可以修改要更新或插入的值。可以将在更新或插入操作前运行的触发器用于下列几种用途:

- 在数据库中实际更新或插入值之前检查或修改这些值。如果需要将用户看到的数据格式变换为某种内部数据库格式，那么这样做很有用。
- 运行用户定义的函数中编写的其他非数据库操作。

**BEFORE DELETE 触发器**

在删除操作前运行。检查值（必要时产生错误）

**后触发器**

在更新、插入或删除操作后运行。可以将在更新或插入操作后运行的触发器用于下列几种用途:

- 更新其他表中的数据。此功能对于保持数据之间的关系或保留审计跟踪信息很有用。
- 针对表或其他表中的其他数据检查。当引用完整性约束不适合或者表检查约束限制仅对当前表进行检查时，此功能对于确保数据完整性很有用。
- 运行用户定义的函数中编写的非数据库操作。在发出警报或更新数据库外的信息时，此功能很有用。

**INSTEAD OF 触发器**

描述如何对视图执行插入、更新和删除操作，这些视图太复杂，以致无法在本机支持这些操作。这种触发器允许应用程序将视图用作所有 SQL 操作（插入、删除、更新和选择）的唯一界面。

## 前触发器

通过使用在更新或插入操作之前运行的触发器，可以在实际修改数据库之前修改要更新或插入的值。这些触发器可用来在需要时将应用程序中的输入（数据的用户视图）变换为内部数据库格式。

这些前触发器还可用来使其他非数据库操作通过用户定义的函数被激活。

BEFORE DELETE 触发器在删除操作之前运行。它们将检查值，必要时还会产生错误。

### 示例

以下示例将定义一个具有复杂缺省值的 DELETE TRIGGER:

```
CREATE TRIGGER trigger1
  BEFORE UPDATE ON table1
  REFERENCING NEW AS N
  WHEN (N.expected_delivery_date IS NULL)
  SET N.expected_delivery_date = N.order_date + 5 days;
```

以下示例将定义一个具有不是引用完整性约束的交叉表约束的 DELETE TRIGGER:

```
CREATE TRIGGER trigger2
  BEFORE UPDATE ON table2
  REFERENCING NEW AS N
  WHEN (n.salary > (SELECT maxsalary FROM salaryguide WHERE rank = n.position))
  SIGNAL SQLSTATE '78000' SET MESSAGE_TEXT = 'Salary out of range');
```

## 后触发器

可以用几种方式使用在更新、插入或删除操作后运行的触发器。

- 触发器可以更新、插入或删除相同表或其他表中的数据。这对于保持数据之间的关系或保留审计跟踪信息很有用。
- 触发器可以针对剩余表或其他表中的数据值检查数据。如果由于引用了此表中的其他行或其他表中的数据而导致无法使用引用完整性约束或检查约束，那么这样做很有用。
- 触发器可以使用用户定义的函数来激活非数据库操作。这样做很有用，例如，用于发出警报或更新数据库外的信息。

### 示例

以下示例显示一个后触发器，它在雇佣新职员时增大职员数。

```
CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

## INSTEAD OF 触发器

INSTEAD OF 触发器描述如何对复杂视图执行插入、更新和删除操作。INSTEAD OF 触发器允许应用程序将视图用作所有 SQL 操作（插入、删除、更新和选择）的唯一界面。

通常，INSTEAD OF 触发器包含视图主体中应用的逻辑的相反逻辑。例如，考虑一个用于解密其源表中的列的视图。此视图的 INSTEAD OF 触发器加密数据，然后将它插入到源表中，因此执行对称操作。

通过使用 INSTEAD OF 触发器，请求对视图执行的修改操作将替换为触发器逻辑，该逻辑代表视图执行操作。从应用程序的角度来看，这是透明地进行的，因为它看到所有操作都是对视图执行的。只允许将一个 INSTEAD OF 触发器用于给定主题视图的每种操作。

视图本身必须是隐式类型视图或解析为隐式类型视图的别名。此外，它不能是使用 WITH CHECK OPTION 定义的视图（对称视图）或在其上直接或间接定义了对称视图的视图。

### 示例

以下示例显示了三个 INSTEAD OF 触发器，它们为已定义的视图（EMPV）提供 INSERT、UPDATE 和 DELETE 逻辑。视图 EMPV 的 FROM 子句中包含连接，因此不能在本机支持任何修改操作。

```
   CREATE VIEW EMPV(EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO,
                    HIREDATE, DEPTNAME)
   AS SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO, HIREDATE, DEPTNAME
FROM EMPLOYEE, DEPARTMENT WHERE EMPLOYEE.WORKDEPT = DEPARTMENT.DEPTNO
   CREATE TRIGGER EMPV_INSERT INSTEAD OF INSERT ON EMPV
   REFERENCING NEW AS NEWEMP FOR EACH ROW
   INSERT INTO EMPLOYEE (EMPNO, FIRSTNME, MIDINIT, LASTNAME,
                         WORKDEPT, PHONENO, HIREDATE)
          VALUES(EMPNO, FIRSTNME, MIDINIT, LASTNAME,
                 COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
                           WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
                          RAISE_ERROR('70001', 'Unknown dept name')),
                 PHONENO, HIREDATE)

   CREATE TRIGGER EMPV_UPDATE INSTEAD OF UPDATE ON EMPV
   REFERENCING NEW AS NEWEMP OLD AS OLDEMP
     FOR EACH ROW
   BEGIN ATOMIC
    VALUES(CASE WHEN NEWEMP.EMPNO = OLDEMP.EMPNO THEN 0
                ELSE RAISE_ERROR('70002', 'Must not change EMPNO') END);
    UPDATE EMPLOYEE AS E
      SET (FIRSTNME, MIDINIT, LASTNAME, WORKDEPT, PHONENO, HIREDATE)
        = (NEWEMP.FIRSTNME, NEWEMP.MIDINIT, NEWEMP.LASTNAME,
                 COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
                           WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
                          RAISE_ERROR ('70001', 'Unknown dept name')),
                 NEWEMP.PHONENO, NEWEMP.HIREDATE)
    WHERE NEWEMP.EMPNO = E.EMPNO;
   END

   CREATE TRIGGER EMPV_DELETE INSTEAD OF DELETE ON EMPV
   REFERENCING OLD AS OLDEMP FOR EACH ROW
   DELETE FROM EMPLOYEE AS E WHERE E.EMPNO = OLDEMP.EMPNO
```

## 设计触发器

创建触发器时，必须将它与表关联；创建 INSTEAD OF 触发器时，必须将它与视图关联。此表或视图称为触发器的*目标表*。术语修改操作指的是目标表状态中的任何更改。

*修改操作*由下列各项启动:

- INSERT 语句
- UPDATE 语句或执行更新的引用约束
- DELETE 语句或执行删除的引用约束
- MERGE 语句

必须将每个触发器与这三种类型的修改操作中的一种操作关联。关联称为该特定触发器的*触发器事件*。

还必须定义发生触发器事件时触发器执行的操作，即*触发操作*。触发操作由一个或多个语句组成，可以在数据库管理器执行触发器事件前后执行这些语句。发生触发器事件后，数据库管理器将确定主题表中受修改操作影响的行集并执行触发器。

**创建触发器时的准则:**

创建触发器时，必须声明下列属性和行为:

- 触发器的名称。
- 主题表的名称。
- 触发器激活时间（在修改操作执行前或执行后）。
- 触发器事件（INSERT、DELETE 或 UPDATE）。
- 旧转换变量值（如果存在）
- 新转换变量值（如果存在）
- 旧转换表值（如果存在）
- 新转换表值（如果存在）
- 粒度（FOR EACH STATEMENT 或 FOR EACH ROW）。
- 触发器的触发操作（包括触发操作条件和触发语句）。
- 触发器事件是 UPDATE 时，如果仅当 UPDATE 语句中指定了特定列时才应触发触发器，那么必须声明触发器列列表。

**设计多个触发器:**

使用 CREATE TRIGGER 语句定义触发器时，触发器的创建时间将以时间戳记格式登记在数据库中。如果有多个应同时运行的触发器时，那么可以在以后使用此时间戳记值来对触发器的激活顺序进行排序。例如，如果具有相同事件和相同激活时间的相同主题表上有多个触发器，那么将使用时间戳记。如果有一个或多个后触发器或 INSTEAD OF 触发器是由触发器事件和触发操作直接或间接（即，通过其他引用约束递归）导致的引用约束操作激活的，那么也可以使用时间戳记。

请考虑下列两个触发器:

```
CREATE TRIGGER NEW_HIRED
        AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
    BEGIN ATOMIC
    UPDATE COMPANY_STATS
    SET NBEMP = NBEMP + 1;
    END
CREATE TRIGGER NEW_HIRED_DEPT
         AFTER INSERT ON EMPLOYEE
   REFERENCING NEW AS EMP
     FOR EACH ROW
       BEGIN ATOMIC
       UPDATE DEPTS
       SET NBEMP = NBEMP + 1
       WHERE DEPT_ID = EMP.DEPT_ID;
       END
```

对 EMPLOYEE 表运行插入操作时将激活上述触发器。在本示例中，触发器的创建时间戳记定义先激活上述两个触发器中的哪个触发器。

按时间戳记值的升序顺序激活触发器。因此，刚刚添加到数据库的触发器在先前定义的所有其他触发器后运行。

旧触发器在新触发器之前激活以确保可以将新触发器用作影响数据库的更改的*递增*增补。例如，如果触发器 T1 的触发语句将一个新行插入到表 T 中，那么可以使用在 T1 后运行的触发器 T2 的触发语句来更新 T 中具有特定值的相同行。因为触发器的激活顺序可预测，所以可以让一个表上有多个触发器，并且还知道较新的触发器将处理已由较旧的触发器修改的表。

**触发器与引用约束交互:**

由于强制执行引用约束而产生的更改可能会导致发生触发器事件。例如，给定两个表 DEPT 和 EMP，如果删除或更新 DEPT 导致通过引用完整性约束将删除或更新传播至 EMP，那么在 EMP 上定义的删除或更新触发器将由于 DEPT 上定义的引用约束而激活。EMP 上的触发器将在删除（在 ON DELETE CASCADE 情况下）或更新 EMP 中的行（在 ON DELETE SET NULL 情况下）之前或之后运行，这取决于触发器的激活时间。

## 指定使触发器触发的对象（触发语句或事件）

每个触发器都与一个事件关联。当数据库中发生触发器的相应事件时，就会激活触发器。对目标表执行指定的操作（UPDATE、INSERT 或 DELETE 语句，包括引用约束的操作所产生的语句）时，就会发生此触发器事件。

例如:

```
CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

以上语句定义触发器 new_hire，当您对表 employee 执行插入操作时，就会激活该触发器。

将每个触发器事件以及每个触发器与一个目标表和一个修改操作关联。修改操作为:

**插入操作**

插入操作只能由 INSERT 语句或 MERGE 语句的插入操作产生。因此，通过未使用 INSERT 的实用程序（例如，LOAD 命令）装入数据时，不会激活触发器。

**删除操作**

删除操作可以由 DELETE 语句或 MERGE 语句的删除操作产生，或者由 ON DELETE CASCADE 的引用约束规则产生。

**更新操作**

更新操作可以由 UPDATE 语句或 MERGE 语句的更新操作产生，或者由 ON DELETE SET NULL 的引用约束规则产生。

如果触发器事件是更新操作，那么该事件可以与目标表的特定列关联。在这种情况下，仅当更新操作尝试更新任何指定列时，才会激活触发器。这将进一步改进激活触发器的事件。

例如，仅当对表 PARTS 的 ON_HAND 或 MAX_STOCKED 列执行更新操作时，以下触发器 REORDER 才激活:

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS N_ROW
    FOR EACH ROW
  WHEN (N_ROW.ON_HAND < 0.10 * N_ROW.MAX_STOCKED)
    BEGIN ATOMIC
  VALUES(ISSUE_SHIP_REQUEST(N_ROW.MAX_STOCKED -
                            N_ROW.ON_HAND,
                            N_ROW.PARTNO));
    END
```

激活触发器时，它将根据其粒度级别运行，如下所示:

**FOR EACH ROW**
它运行的次数与受影响的行集中的行数相同。如果需要引用触发操作所影响的特定行，请使用 FOR EACH ROW 粒度。这种示例比较 AFTER UPDATE 触发器中已更新行的新值和旧值。

**FOR EACH STATEMENT**
它对整个触发器事件运行一次。

如果受影响的行集为空（也就是说，当搜索式 UPDATE 或 DELETE 中的 WHERE 子句未限定任何行时），FOR EACH ROW 触发器不运行。但 FOR EACH STATEMENT 触发器仍运行一次。

例如，可以使用 FOR EACH ROW 来计算职员数。

```
CREATE TRIGGER NEW_HIRED
        AFTER INSERT ON EMPLOYEE
  FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

通过使用粒度 FOR EACH STATEMENT 并执行一个更新操作可以获得相同的效果。

```
CREATE TRIGGER NEW_HIRED
        AFTER INSERT ON EMPLOYEE
  REFERENCING NEW_TABLE AS NEWEMPS
  FOR EACH STATEMENT
  UPDATE COMPANY_STATS
  SET NBEMP = NBEMP + (SELECT COUNT(*) FROM NEWEMPS)
```

**注:**

- 前触发器不支持 FOR EACH STATEMENT 粒度。
- 触发器的最大嵌套级别为 16。即，级联触发器激活的最大数目为 16。触发器激活指的是发生触发事件（如插入、更新或删除表或表列中的数据）时激活触发器。

## 指定触发器触发的时间（BEFORE、AFTER 和 INSTEAD OF 子句）

*触发器激活时间*指定与触发器事件相比，应激活触发器的时间。

您可以指定三个激活时间: BEFORE、AFTER 或 INSTEAD OF:

- 如果激活时间是 BEFORE，那么将在触发器事件执行之前对受影响的行集中的每行激活触发操作。因此，只有在前触发器完成每行的执行后才修改主题表。请注意，前触发器必须具有 FOR EACH ROW 粒度。
- 如果激活时间是 AFTER，那么将对受影响的行集中的每行或对语句激活触发操作，这取决于触发器粒度。此操作在触发器事件完成后并且在数据库管理器检查触发器事件可能影响的所有约束（包括引用约束的操作）后发生。请注意，后触发器可以具有 FOR EACH ROW 或 FOR EACH STATEMENT 粒度。

  例如，以下触发器的激活时间是在对 employee 执行插入操作后:

```
CREATE TRIGGER NEW_HIRE
  AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP + 1
```

- 如果激活时间是 INSTEAD OF，那么将对受影响的行集中的每行激活触发操作，而不是执行触发器事件。INSTEAD OF 触发器必须具有 FOR EACH ROW 粒度，并且主题表必须是视图。其他触发器均无法将视图用作主题表。

下图说明了前触发器和后触发器的执行模型:

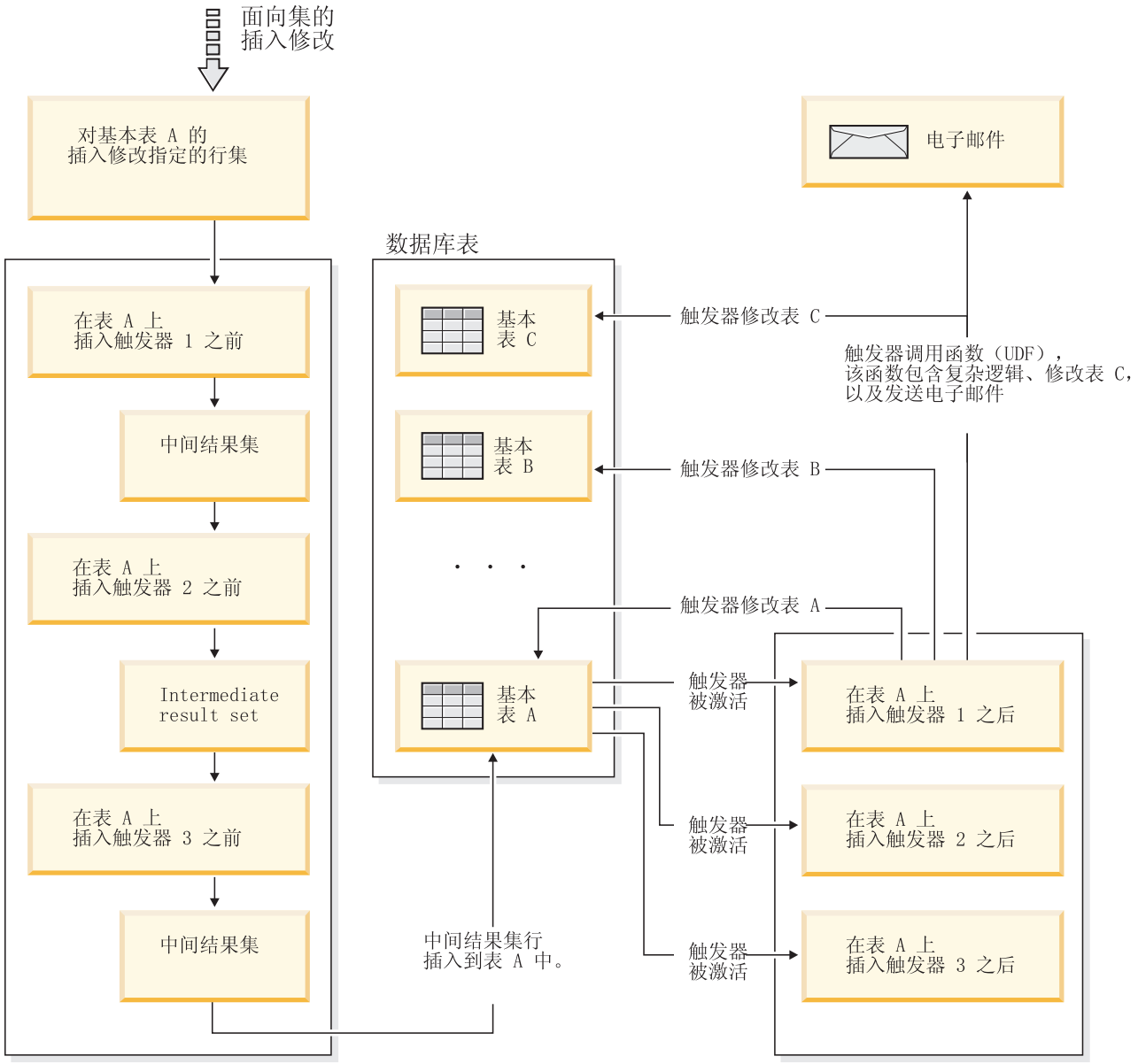



*图 26. 触发器执行模型*

对于同时具有前触发器和后触发器的给定表，以及与这些触发器关联的修改事件，将首先激活所有前触发器。首先为给定事件激活的前触发器将处理操作的目标行集，并对该行集进行其逻辑限定的任何更新修改。此前触发器的输出由下一个前触发器作为输入接受。当触发了事件激活的所有前触发器时，中间结果集（即触发器事件操作的目标行进行的前触发器修改的结果）将应用于基本表。然后，将触发与该事件关联的每个后触发器。后触发器可以修改同一个表、另一个表或在数据库外部执行操作。

不同的触发器激活时间反映不同的触发器用途。基本上，前触发器是对数据库管理系统的约束子系统的扩展。因此，您通常使用它们来:

- 验证输入数据
- 自动生成新插入的行的值
- 为交叉引用而从其他表中进行读取

由于前触发器是在将触发器事件应用于数据库之前激活的，所以不使用它们来进一步修改数据库。因此，在检查完整性约束之前激活这些触发器。

相反，可以将后触发器视为每次特定事件发生时就在数据库中运行的应用程序逻辑的模块。作为应用程序的一部分，后触发器始终看到处于一致状态的数据库。请注意，它们在完整性约束验证后运行。因此，主要将它们用来执行应用程序也可以执行的操作。例如:

- 在数据库中继续执行修改操作。
- 在数据库外执行操作，例如，用以支持警报。请注意，回滚触发器时不会回滚在数据库外执行的操作。

比较而言，可以将 INSTEAD OF 触发器视为对定义该触发器的视图的反向操作的描述。例如，如果视图中的选择列表包含一个基于表的表达式，那么其 INSTEAD OF INSERT 触发器的主体中的 INSERT 语句将包含反向表达式。

因为前触发器、后触发器和 INSTEAD OF 触发器具有不同的性质，所以可以使用一组不同的 SQL 操作来定义前触发器、后触发器和 INSTEAD OF 触发器的触发操作。例如，前触发器中不允许更新操作，这是因为不能保证触发操作不会违反完整性约束。同样，前触发器、后触发器和 INSTEAD OF 触发器中支持不同的触发器粒度。

所有触发器的触发 SQL 语句可以是动态复合语句。但是，前触发器会受到一些限制; 它们不能包含下列 SQL 语句:

- UPDATE
- DELETE
- INSERT
- MERGE

## 定义触发器操作将触发的条件（WHEN 子句）

激活触发器将导致运行与该触发器关联的触发操作。每个触发器正好有一个触发操作，而该触发操作又有两个组件: 可选的*触发操作条件*或 WHEN 子句以及*触发语句*集。

*触发操作条件*是触发操作的可选子句，它指定一个搜索条件，必须对该条件求值为 true 才能运行触发操作内的语句。如果省略 WHEN 子句，那么始终执行触发操作内的语句。

如果触发器是 FOR EACH ROW 触发器，那么将针对每行对触发操作条件进行一次求值; 如果触发器是 FOR EACH STATEMENT 触发器，那么将针对每个语句对该条件进行一次求值。

此子句提供了进一步的控制能力，您可以用来代表触发器微调激活的操作。体现 WHEN 子句的用处的一个示例是强制执行数据从属规则，在该规则中，仅当入局值在某些范围内或某个范围外时，才激活触发操作。

激活触发器将导致运行与该触发器关联的触发操作。每个触发器正好有一个触发操作，而该触发操作又有两个组件:

触发操作条件定义是针对正在对其执行触发操作的行还是语句执行触发语句集。触发语句集定义由于发生触发器事件而由触发器在数据库中执行的操作集。

例如，以下触发器操作指定只应对 on_hand 列的值小于 max_stocked 列值的 10% 的行激活触发语句集。在本示例中，触发语句集是调用 issue_ship_request 函数。

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS N_ROW
    FOR EACH ROW

  WHEN (N_ROW.ON_HAND < 0.10 * N_ROW.MAX_STOCKED)
    BEGIN ATOMIC
    VALUES(ISSUE_SHIP_REQUEST(N_ROW.MAX_STOCKED -
                              N_ROW.ON_HAND,
                              N_ROW.PARTNO));
  END
```

触发语句集执行激活触发器所产生的实际操作。并非每个 SQL 操作在每个触发器中都有意义。根据触发器激活时间是 BEFORE 还是 AFTER，不同类型的操作可能适合用作触发语句。

在大多数情况下，如果任何触发语句返回负返回码，那么将回滚触发语句以及所有触发器和引用约束操作。错误消息中将返回触发器名、SQLCODE、SQLSTATE 和失败的触发语句中的许多标记。

## 触发器中受支持的 SQL PL 语句

所有触发器的触发 SQL 语句可以是动态复合语句。

也就是说，触发 SQL 语句可以包含下列一个或多个元素:

- CALL 语句
- DECLARE Variable 语句
- SET Variable 语句
- WHILE 循环
- FOR 循环
- IF 语句
- SIGNAL 语句
- ITERATE 语句
- LEAVE 语句
- GET DIGNOSTIC 语句
- 全查询

但是，只有后触发器和 INSTEAD OF 触发器可以包含下列一个或多个 SQL 语句:

- UPDATE 语句
- DELETE 语句
- INSERT 语句
- MERGE 语句

## 使用转换变量访问触发器中的旧列值和新列值

实施 FOR EACH ROW 触发器时，可能需要引用该触发器当前正在对其执行的受影响行集中的行的列值。请注意，要引用数据库表（包括主题表）中的列，可以使用一般 SELECT 语句。

通过使用可以在 CREATE TRIGGER 语句的 REFERENCING 子句中指定的两个转换变量，FOR EACH ROW 触发器可以引用它当前正在对其执行的行的列。有两种类型的转换变量，它们被指定为 OLD 和 NEW 并带有 correlation-name。它们具有下列语义:

**OLD AS correlation-name**

指定一个相关名，它捕获行的原始状态（即，在将触发操作应用于数据库之前）。

**NEW AS correlation-name**

指定一个相关名，它捕获在将触发操作应用于数据库时用于更新该数据库中的行的值。

请考虑以下示例:

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW AS N_ROW
  FOR EACH ROW
  WHEN (N_ROW.ON_HAND < 0.10 * N_ROW.MAX_STOCKED
  AND N_ROW.ORDER_PENDING = 'N')
    BEGIN ATOMIC
    VALUES(ISSUE_SHIP_REQUEST(N_ROW.MAX_STOCKED -
                              N_ROW.ON_HAND,
                              N_ROW.PARTNO));
    UPDATE PARTS SET PARTS.ORDER_PENDING = 'Y'
    WHERE PARTS.PARTNO = N_ROW.PARTNO;
    END
```

根据上面给出的 OLD 和 NEW 转换变量的定义，很明显并不能对每个触发器都定义每个转换变量。可以根据触发器事件的类型来定义转换变量:

**更新** 更新触发器可以同时引用 OLD 和 NEW 转换变量。

**插入** 由于在激活插入操作之前数据库中不存在受影响的行，所以插入触发器只能引用 NEW 转换变量。也就是说，在将触发操作应用于数据库之前，没有将定义旧值的行的原始状态。

**删除** 由于在删除操作中未指定新值，所以删除触发器只能引用 OLD 转换变量。

**注**: 只能对 FOR EACH ROW 触发器指定转换变量。在 FOR EACH STATEMENT 触发器中，要指定转换变量正在引用受影响行集中的哪些行，只引用该转换变量并不够。使用 CREATE TRIGGER 语句的 OLD TABLE 和 NEW TABLE 子句来引用新行集和旧行集。有关这些子句的更多信息，请参阅 CREATE TRIGGER 语句。

## 使用转换表引用旧表结果集和新表结果集

在 FOR EACH ROW 和 FOR EACH STATEMENT 触发器中，可能需要引用整个受影响的行集。例如，当触发器主体需要对受影响的行集（例如，某些列值的 MAX、MIN 或 AVG）应用聚集时，就需要引用整个受影响的行集。

通过使用在 CREATE TRIGGER 语句的 REFERENCING 子句中指定的两个转换表，触发器可以引用受影响的行集。与转换变量一样，有两种类型的转换表，使用下列语义将它们指定为 OLD_TABLE 和 NEW_TABLE 并带有 table-name:

**OLD_TABLE AS table-name**

指定一个表名，该表捕获受影响行集的原始状态（即，在将触发 SQL 操作应用于数据库之前）。

**NEW_TABLE AS table-name**

指定一个表名，该表捕获在将触发操作应用于数据库时用于更新数据库中的行的值。

例如:

```
CREATE TRIGGER REORDER
  AFTER UPDATE OF ON_HAND, MAX_STOCKED ON PARTS
  REFERENCING NEW_TABLE AS N_TABLE
  NEW AS N_ROW
    FOR EACH ROW
  WHEN ((SELECT AVG (ON_HAND) FROM N_TABLE) > 35)
    BEGIN ATOMIC
    VALUES(INFORM_SUPERVISOR(N_ROW.PARTNO,
                             N_ROW.MAX_STOCKED,
                             N_ROW.ON_HAND));
    END
```

请注意，NEW_TABLE 始终包含完整的已更新行集，即使对于 FOR EACH ROW 触发器亦如此。触发器对定义它的表进行操作时，NEW_TABLE 将包含从激活该触发器的语句开始的已更改行。但是，NEW_TABLE 不包含由该触发器内的语句产生的已更改行，因为这样会导致单独激活触发器。

转换表是只读表。用于定义可以对触发器事件定义的转换变量类型的相同规则适用于转换表:

**更新** 更新触发器可以引用 OLD_TABLE 和 NEW_TABLE 转换表。

**插入** 由于在激活插入操作之前数据库中不存在受影响的行，所以插入触发器只能引用 NEW_TABLE 转换表。也就是说，在将触发操作应用于数据库之前，没有定义旧值的行的原始状态。

**删除** 由于在删除操作中未指定新值，所以删除触发器只能引用 OLD_TABLE 转换表。

**注:** 了解可以对后触发器的两种粒度 FOR EACH ROW 和 FOR EACH STATEMENT 指定转换表非常重要。

OLD_TABLE 和 NEW_TABLE table-name 的作用域是触发器主体。在此作用域中，此名称优先于具有模式中可能存在的相同未限定 *table-name* 的任何其他表名。因此，如果 OLD_TABLE 或 NEW_TABLE table-name 是 X（此处是举例说明），那么在 SELECT 语句的 FROM 子句中引用 X（即，未限定的 X）将始终引用转换表，即使触发器创建程序的模式中有一个名为 X 的表亦如此。在这种情况下，用户必须使用标准名称才能引用模式中的表 X。

## 创建触发器

触发器定义一组操作，这组操作与用于指定表或类型表的 INSERT、UPDATE 或 DELETE 子句一起执行或由这些子句触发。

使用触发器来执行下列操作:

- 验证输入数据
- 为新插入的行生成值
- 为交叉引用而从其他表中进行读取
- 为审计跟踪而向其他表写入

可使用触发器支持一般形式的完整性或业务规则。例如，在接受订单或更新摘要数据表之前，触发器可以检查客户的信用额度。

**优点**:

- 更快地开发应用程序: 因为触发器存储在数据库中，所以不必编写触发器在每个应用程序中执行的操作。
- 更容易维护: 一旦定义了一个触发器，那么当访问创建它所基于的表时，会自动调用该触发器。
- 业务规则的全局实现: 如果业务策略改变，只需更改触发器而不必更改每个应用程序。

**限制**:

- 不能使用具有昵称的触发器。
- 如果触发器是一个前触发器，那么由触发操作指定的列名不能是除标识列外的生成列。即，生成的标识值对前触发器可视。

当创建原子触发器时，必须认真对待语句结束字符。缺省情况下，命令行处理器将“;”当作是语句结束标记。您应该在脚本中手动编辑语句结束字符来创建原子触发器，以便使用一个非“;”的字符。例如，可以用另一个特殊字符（如“#”）替换“;”。还可以在 CREATE TRIGGER DDL 前面加上下列内容:

```
--#SET TERMINATOR @
```

要更改正在处理的 CLP 中的终止符，以下语法将复原它:

```
--#SET TERMINATOR
```

要通过命令行来创建触发器，请输入:

```
db2 -td <delimiter> -vf <script>
```

其中 <delimiter> 是备用语句结束字符，而 <script> 是使用新 <delimiter> 的已修改脚本。

要通过命令行来创建触发器，请输入:

```
CREATE TRIGGER <name>
      <action> ON <table_name>
      <operation>
<triggered_action>
```

下列语句创建一个触发器，它在每次雇佣新人时会增加职员数，方法为每次向 EMPLOYEE 表添加一行时就在 COMPANY_STATS 表的职员数（NBEMP）列中加 1。

```
CREATE TRIGGER NEW_HIRED
        AFTER INSERT ON EMPLOYEE
    FOR EACH ROW
  UPDATE COMPANY_STATS SET NBEMP = NBEMP+1;
```

触发器主体可以包括下列一个或多个语句：INSERT、搜索式 UPDATE、搜索式 DELETE、全查询、SET Variable 和 SIGNAL SQLSTATE。可以在触发器引用的 INSERT、UPDATE 或 DELETE 语句之前或之后激活触发器。

## 修改和删除触发器

不能修改触发器。必须删除触发器，然后根据您需要的新定义再次进行创建。

**触发器依赖性**

- 触发器与某个其他对象的所有依赖性都记录在 SYSCAT.TRIGDEP 系统目录视图中。一个触发器可依赖许多个对象。
- 如果触发器所依赖的某个对象被删除，那么该触发器就会不可用，但它的定义仍保留在系统目录视图中。要重新验证此触发器，必须从系统目录视图中检索它的定义并提交新的 CREATE TRIGGER 语句。
- 如果删除触发器，那么它的描述会从 SYSCAT.TRIGGERS 系统目录视图中被删除，且它的所有依赖性也会从 SYSCAT.TRIGDEP 系统目录视图中被删除。所有与该触发器有 UPDATE、INSERT 或 DELETE 关系的程序包都会变得无效。
- 如果视图从属于触发器且已使该视图不可用，那么触发器也会标记为不可用。任何从属于已标记为不可用的触发器的程序包都会变得无效。

可以使用 DROP TRIGGER 语句删除触发器对象，但是此过程将导致从属程序包被标记为无效，如下所示:

- 如果删除不带显式列列表的更新触发器，那么对目标表起更新作用的程序包将变得无效。
- 如果删除带一个列列表的更新触发器，那么仅当该程序包也可更新 CREATE TRIGGER 语句的列名列表中至少一列时，用于更新目标表的该程序包才变得无效。
- 如果删除插入触发器，那么用于插入目标表的程序包将变得无效。
- 如果删除删除触发器，那么用于删除目标表的程序包将变得无效。

程序包将保持无效，直到显式绑定或重新绑定该应用程序，或运行它且数据库管理器自动重新绑定它为止。

## 触发器和触发器用法的示例

### 触发器与引用约束之间的交互的示例

更新操作可能会导致触发器与引用约束和检查约束交互。

第 253 页的图 23 和相关描述是对更新数据库中的数据的语句所执行的具有代表性的处理。

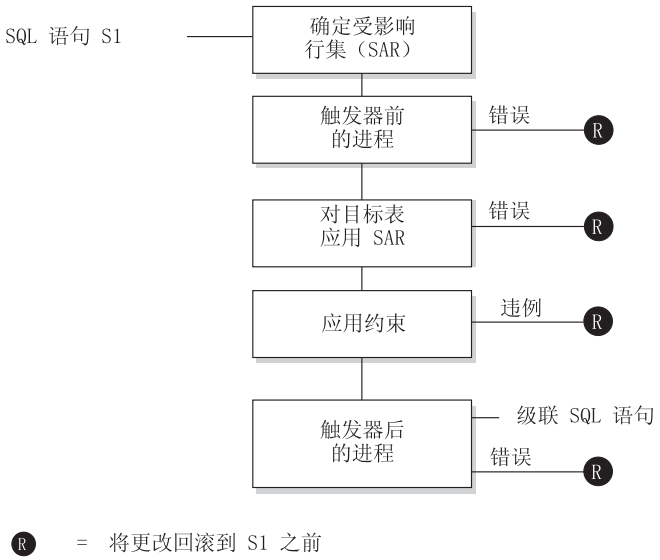



*图 27. 处理包含相关触发器和约束的语句*

第 253 页的图 23 显示用于处理更新表的语句的一般顺序。假定表包含级联的前触发器、引用约束、检查约束和后触发器。以下是对第 253 页的图 23 中的框和其他项的描述。

- 语句 $S_1$

  这是开始过程的 DELETE、INSERT 或 UPDATE 语句。在此描述中，语句 $S_1$ 标识一个称为*主题表*的表（或基于某个表的可更新视图）。

- 确定受影响行集

  此步骤是对 CASCADE 和 SET NULL 的引用约束删除规则以及对后触发器中的级联语句重复的过程的起始点。

  此步骤的用途是确定语句的*受影响行集*。包括的行集基于语句:

  – 对于 DELETE，受影响行集是符合语句的搜索条件的所有行（或定位 DELETE 的当前行）
  – 对于 INSERT，受影响行集是由 VALUES 子句或全查询标识的行
  – 对于 UPDATE，受影响行集是符合搜索条件的所有行（或定位 UPDATE 的当前行）。

  如果受影响行集为空，那么将没有前触发器、没有适用于主题表的更改或没有要对语句处理的约束。

- 处理前触发器

  按创建日期的升序顺序处理所有前触发器。每个前触发器将对受影响行集中的每行处理一次触发操作。

在处理触发操作期间可能会发生错误，在这种情况下，由于原始语句 $S_1$ 产生的所有更改（到目前为止的情况）都将回滚。

如果没有前触发器或者受影响行集为空，那么将跳过此步骤。

- 将受影响行集应用于主题表

  使用受影响行集将实际删除、插入或更新操作应用于数据库中的主题表。

  应用受影响行集时可能会发生错误（例如，在唯一索引存在的情况下尝试插入具有重复键的行），在这种情况，由于原始语句 $S_1$ 产生的所有更改（到目前为止的情况）都将回滚。
- 应用约束

  如果受影响行集不为空，那么将应用与主题表关联的约束。这包括唯一约束、唯一索引、引用约束、检查约束和与视图上的 WITH CHECK OPTION 相关的检查。带有 CASCADE 或 SET NULL 删除规则的引用约束可能导致激活其他触发器。

  违反任何约束或 WITH CHECK OPTION 将产生错误，并且由于 $S_1$ 产生的所有更改（到目前为止的情况）都将回滚。
- 处理后触发器

  按创建日期的升序顺序处理 $S_1$ 激活的所有后触发器。

  即使受影响行集为空，FOR EACH STATEMENT 触发器也正好处理一次触发操作。FOR EACH ROW 触发器将对受影响行集中的每行处理一次触发操作。

  在处理触发操作期间可能会发生错误，在这种情况下，由于原始语句 $S_1$ 产生的所有更改（到目前为止的情况）都将回滚。

  触发器的触发操作可能包括一些触发语句，例如，DELETE、INSERT 或 UPDATE 语句。在此描述中，将每个这种语句都视为*级联语句*。

  级联语句是在后触发器的触发操作中处理的 DELETE、INSERT 或 UPDATE 语句。此语句开始级联级触发器处理。可以将此过程视为将触发语句指定为新的 $S_1$，并循环执行此处描述的所有步骤。

  处理完每个 $S_1$ 激活的所有后触发器中的所有触发语句后，对原始 $S_1$ 的处理就完成了。
- R = 将更改回滚到 $S_1$ 之前

  处理期间发生的任何错误（包括约束违例）将导致回滚由于原始语句 $S_1$ 直接或间接产生的所有更改。因此，数据库将返回到正好在执行原始语句 $S_1$ 之前所处的那个状态。

## 使用触发器定义操作的示例

假定总经理要将在过去 72 小时内发送了三个或更多个投诉的客户名保存在一个单独的表中。总经理还希望在一个客户名多次插入到此表中时通知他/她。

要定义这种操作，请定义:

- 一个 UNHAPPY_CUSTOMERS 表:

```
CREATE TABLE UNHAPPY_CUSTOMERS (
  NAME           VARCHAR (30),
  EMAIL_ADDRESS  VARCHAR (200),
  INSERTION_DATE DATE)
```

- 一个触发器，它用于在过去 3 天内接收到 3 条或更多条消息时自动在 UNHAPPY_CUSTOMERS 中插入一行（假定存在一个 CUSTOMERS 表，它包括 NAME 列和 E_MAIL_ADDRESS 列）：

```
CREATE TRIGGER STORE_UNHAPPY_CUST
  AFTER INSERT ON ELECTRONIC_MAIL
    REFERENCING NEW AS N
    FOR EACH ROW
  WHEN (3 <= (SELECT COUNT(*)
              FROM ELECTRONIC_MAIL
              WHERE SENDER = N.SENDER
                AND SENDING_DATE(MESSAGE) > CURRENT DATE - 3 DAYS)
      )
    BEGIN ATOMIC
    INSERT INTO UNHAPPY_CUSTOMERS
    VALUES ((SELECT NAME
    FROM CUSTOMERS
    WHERE EMAIL_ADDRESS = N.SENDER), N.SENDER, CURRENT DATE);
    END
```

- 一个触发器，它用于在同一客户多次插入到 UNHAPPY_CUSTOMERS 中时向总经理发送通知（假定存在 SEND_NOTE 函数，它使用 2 个字符串作为输入）：

```
CREATE TRIGGER INFORM_GEN_MGR
  AFTER INSERT ON UNHAPPY_CUSTOMERS
    REFERENCING NEW AS N
    FOR EACH ROW
  WHEN (1 <(SELECT COUNT(*)
            FROM UNHAPPY_CUSTOMERS
            WHERE EMAIL_ADDRESS = N.EMAIL_ADDRESS)
      )
    BEGIN ATOMIC
    VALUES(SEND_NOTE('Check customer:' CONCAT N.NAME,
                     'bigboss@vnet.ibm.com'));
    END
```

## 使用触发器定义业务规则的示例

假定您所在的公司有一个策略，要求所有处理客户投诉的电子邮件都必须将市场部经理 Mr. Nelson 包括在副本（CC）列表中。

由于这是规则，所以您可能要将它表示成诸如下列其中一个约束（假定存在 CC_LIST UDF 以进行检查）：

```
ALTER TABLE ELECTRONIC_MAIL ADD
  CHECK (SUBJECT <> 'Customer complaint' OR
         CONTAINS (CC_LIST(MESSAGE), 'nelson@vnet.ibm.com') = 1)
```

但是，这种约束不允许插入处理客户投诉但 CC 列表中未包括市场部经理的电子邮件。这当然不是公司业务规则的意图。公司的意图是将任何处理客户投诉但未复制给市场部经理的电子邮件转发给市场部经理。这种业务规则只能用触发器来表示，因为它需要执行操作，而这无法通过声明约束来表示。触发器假定存在 SEND_NOTE 函数且参数类型为 E_MAIL 和字符串。

```
CREATE TRIGGER INFORM_MANAGER
  AFTER INSERT ON ELECTRONIC_MAIL
    REFERENCING NEW AS N
    FOR EACH ROW
  WHEN (N.SUBJECT = 'Customer complaint' AND
```

```
CONTAINS (CC_LIST(MESSAGE), 'nelson@vnet.ibm.com') = 0)
BEGIN ATOMIC
VALUES(SEND_NOTE(N.MESSAGE, 'nelson@vnet.ibm.com'));
END
```

## 使用触发器防止对表进行操作的示例

假定您要防止无法投递的电子邮件存储在名为 ELECTRONIC_MAIL 的表中。为此，您需要防止执行某些 SQL INSERT 语句。

可使用两种方法来实现此目的:

- 定义一个前触发器，它在电子邮件主题为 *undelivered mail* 时返回错误:

```
CREATE TRIGGER BLOCK_INSERT
  NO CASCADE BEFORE INSERT ON ELECTRONIC_MAIL
    REFERENCING NEW AS N
    FOR EACH ROW
  WHEN (SUBJECT(N.MESSAGE) = 'undelivered mail')
    BEGIN ATOMIC
    SIGNAL SQLSTATE '85101'
      SET MESSAGE_TEXT = ('Attempt to insert undelivered mail');
       END
```

- 定义一个检查约束，它强制使新列 SUBJECT 的值不同于 *undelivered mail*:

```
ALTER TABLE ELECTRONIC_MAIL
  ADD CONSTRAINT NO_UNDELIVERED
  CHECK (SUBJECT <> 'undelivered mail')
```

# 第 15 章 序列

*序列*是一个数据库对象，它允许自动生成值，例如，支票号。序列特别适合于生成唯一键值这一任务。应用程序可以使用序列来避免用于跟踪数字的列值所引起的可能的并行性和性能问题。与在数据库外部创建的数字相比，序列的优点在于数据库服务器将跟踪生成的数字。崩溃和重新启动不会导致生成重复的数字。

生成的序号具有下列属性:

- 值可以是小数位为零的任何精确数字数据类型。这样的数据类型包括: SMALLINT、BIGINT、INTEGER 和 DECIMAL。
- 连续值之间可以有任何指定的整数增量。缺省递增值是 1。
- 计数器值是可恢复的。当需要恢复时，从日志中重建计数器值。
- 可以高速缓存值以改善性能。在高速缓存中预分配并存储值，可以在为序列生成值时减少对日志的同步 I/O。在系统出现故障时，将认为尚未使用的所有高速缓存值已丢失。为 CACHE 指定的值是可能丢失的序列值的最大数目。

有两种表达式可与序列一起使用:

- **NEXT VALUE 表达式**: 它对指定序列返回下一个值。当 NEXT VALUE 表达式指定序列的名称时，将生成一个新的序号。但是，如果一个查询中有多个 NEXT VALUE 表达式的实例指定同一序列名，那么对于结果的每一行，序列计数器仅递增一次，且 NEXT VALUE 的所有实例对结果的每一行返回同一个值。
- **PREVIOUS VALUE 表达式**: 对于当前应用程序进程中的先前语句，该表达式对指定序列返回最新生成的值。也就是说，对于任何给定连接，PREVIOUS VALUE 将保持不变，即使另一个连接调用 NEXT VALUE 也是如此。

有关这些表达式的完整详细信息和示例，请参阅*SQL Reference, Volume 1*中的『序列引用』。

## 设计序列

设计序列时，需要考虑标识列与序列之间的差别，以及哪个更适合您的环境。如果决定使用序列，那么您需要熟悉可用的选项和参数。

在设计序列之前，请参阅第 295 页的『比较序列与标识列』。

除了容易设计和创建外，序列还具有其他各种选项，它们允许您更灵活地生成值:

- 从各种数据类型（SMALLINT、INTEGER、BIGINT 或 DECIMAL）中选择
- 更改起始值（START WITH）
- 更改序列增量，包括指定不断增大或不断减小的值（INCREMENT BY）
- 设置最小值和最大值，即序号的起始值和结束值（MINVALUE/MAXVALUE）
- 允许回绕值以便序列可以再次重新开始，或者禁止循环（CYCLE/NO CYCLE）
- 允许高速缓存序列值以提高性能，或者禁止高速缓存（CACHE/NO CACHE）

© Copyright IBM Corp. 1993, 2010

即使在生成序列后，这些值中的许多值也可以改变。例如，您可能要根据星期几来设置另一个起始值。使用序列的另一个实际示例是生成和处理银行支票。银行支票号序列非常重要，如果一组序号丢失或损坏，那么将会产生严重后果。

为了提高性能，还应该了解并使用 CACHE 选项。此选项告知数据库管理器在系统生成多少个序列值后，才返回到目录以生成另一组序列。如果未指定 CACHE 值，那么缺省值为 20。以缺省值为例，在请求第一个序列值时，数据库管理器将自动在内存中生成 20 个连续值（1, 2, ...., 20）。每次需要新的序号时，就会使用此内存高速缓存值来返回下一个值。用完此高速缓存值后，数据库管理器将生成下一组 21 个值（21, 22, ...., 40）。

通过实现序号的高速缓存，数据库管理器不必始终转至目录表来获取下一个值。这将减少与检索序号相关的开销，但在系统出现故障或系统关闭时，它可能还会导致序列中出现间隔。例如，如果您决定将序列高速缓存设置为 100，那么数据库管理器将高速缓存 100 个这样的数字值，并且还将设置系统目录以表明下一个序列值应从 201 开始。在数据库关闭时，下一组序号将从 201 开始。如果未使用生成的从 101 到 200 的数字，那么这些数字将从序列集中丢失。如果您的应用程序无法容忍生成的值中出现间隔，那么需要将高速缓存值设置为 NO CACHE，即使这样会产生较高的系统开销也应如此。

有关所有可用的选项和关联值的更多信息，请参阅 CREATE SEQUENCE 语句。

## 管理序列行为

可以通过调整序列的行为来满足应用程序要求。在发出 CREATE SEQUENCE 语句以创建新序列或对现有序列发出 ALTER SEQUENCE 语句时，可以更改序列的属性。

以下是您可以指定的一些序列属性:

**数据类型**

CREATE SEQUENCE 语句的 AS 子句指定序列的数字数据类型。数据类型确定序列的可能最小值和最大值。SQL 和 XML 限制中列示了数据类型的最小值和最大值。不能更改序列的数据类型；而是必须通过发出 DROP SEQUENCE 语句来删除序列，然后发出带有新数据类型的 CREATE SEQUENCE 语句。

**起始值** CREATE SEQUENCE 语句的 START WITH 子句设置序列的初始值。ALTER SEQUENCE 语句的 RESTART WITH 子句将序列值重新设置为指定的值。

**最小值** MINVALUE 子句设置序列的最小值。

**最大值** MAXVALUE 子句设置序列的最大值。

**增量值** INCREMENT BY 子句设置每个 NEXT VALUE 表达式添加至序列当前值的值。要使序列值递减，请指定一个负数值。

**序列循环**

CYCLE 子句导致达到其最大值或最小值的序列值在随后的 NEXT VALUE 表达式中生成其各自的最小值或最大值。

**注:** 只有在不需要唯一数字或者可以保证在序列循环后不再使用较旧的序列值时，才应该使用 CYCLE。

例如，如果要创建一个名为 `id_values` 的序列，其最小值为 0、最大值为 1000、使用每个 NEXT VALUE 表达式使值递增 2，并且在达到最大值时返回到其最小值，那么请发出以下语句:

```
CREATE SEQUENCE id_values
  START WITH 0
  INCREMENT BY 2
  MAXVALUE 1000
  CYCLE
```

## 应用程序性能和序列

与其他方法相比，使用序列来生成值通常会提高应用程序的性能，这一点与标识列相同。序列的替代方法是创建存储当前值的单列表并使用触发器或在应用程序控制下递增值。但是，在一个分布式环境中，如果应用程序当前访问单列表，那么强制对该表进行序列化访问所需的锁定可能会严重影响性能。

使用序列可以避免与单列表方法关联的锁定问题，并且可以将序列值高速缓存在内存中以减少响应时间。为了让使用序列的应用程序的性能最大，请确保序列高速缓存适当数量的序列值。CREATE SEQUENCE 和 ALTER SEQUENCE 语句的 CACHE 子句指定数据库管理器生成并存储在内存中的最大数目的序列值。

如果序列必须按顺序生成值，并且不会由于系统故障或数据库取消激活而在该顺序中引入间隔，请在 CREATE SEQUENCE 语句中使用 ORDER 和 NO CACHE 子句。NO CACHE 子句保证生成的值中没有间隔，但会使应用程序性能降级一些，因为它每次生成新值时，都会强制将序列写入数据库日志。请注意，由于事务回滚并且未真正使用它们请求的该序列值，所以仍然会出现间隔。

## 比较序列与标识列

虽然对于 DB2 应用程序来说，序列和标识列用途相似，但它们之间存在一个重要差别。标识列使用 LOAD 实用程序自动生成单个表中的列值。序列根据请求使用 CREATE SEQUENCE 语句生成可在任何 SQL 语句中使用的顺序值。

**标识列** 允许数据库管理器自动为添加至表的每一行生成唯一数字值。如果您正在创建一个表并且知道需要唯一标识将添加至该表的每一行，那么可通过 CREATE TABLE 语句向该表定义添加一个标识列。

```
CREATE TABLE <table name>
  (<column name 1> INT,
   <column name 2>, DOUBLE,
   <column name 3> INT NOT NULL GENERATED ALWAYS AS IDENTITY
     (START WITH <value 1>, INCREMENT BY <value 2>))
```

在本示例中，第三列标识标识列。可以定义的其中一个属性是在添加行时用来唯一定义每一行的列中使用的值。INCREMENT BY 子句后面的值显示对于添加至该表的每一行来说标识列内容的后续值的增量。

创建标识属性后，可以使用 ALTER TABLE 语句更改或除去这些属性。还可以使用 ALTER TABLE 语句在其他列中添加标识属性。

**序列** 允许自动生成值。序列特别适合于生成唯一键值这一任务。应用程序可以使用序列来避免通过其他方法生成唯一计数器所引起的可能的并行性和性能问题。与标识列不同，未使序列与特定表列相关，也未将它绑定至唯一表列，只是仅可通过该表列访问。

可以创建序列并在以后改变它，以便它通过无限递增或递减值来生成值；或者递增或递减至用户定义的限制，然后停止；或者递增或递减至用户定义的限制，然后循环回至起点并重新开始。序列仅在单分区数据库中受支持。

以下示例显示如何创建一个名为 orderseq 的序列:

```
CREATE SEQUENCE orderseq
        START WITH 1
        INCREMENT BY 1
        NOMAXVALUE
        NOCYCLE
  CACHE 50
```

在本示例中，序列从 1 开始，并以 1 为增量增加，且没有上限。由于没有指定上限，所以没有理由循环回至起点并从 1 重新开始。CACHE 参数指定了数据库管理器预分配并保存在内存中的序列值的最大数目。

## 创建序列

要创建序列，请使用 CREATE SEQUENCE 语句。与标识列属性不同，未使序列与特定表列相关，也未将它绑定至唯一表列，只是仅可通过该表列访问。

在可使用 NEXT VALUE 或 PREVIOUS VALUE 表达式的位置有几个限制。可以创建或改变序列，以便序列以下列其中一种方式来生成值:

- 单调地递增或递减（按常量更改）且没有限制
- 单调地递增或递减至用户定义的限制并停止
- 单调地递增或递减至用户定义的限制并循环回至起点，然后重新开始。

**注: 在恢复使用序列的数据库时请务必小心**: 对于在数据库外部使用的序列值（例如，用于银行支票的序号），如果将数据库恢复至数据库失败前的一个时间点，那么可能会导致对某些序列生成重复值。要避免可能的重复值，不应该将在数据库外部使用序列值的数据库恢复至前一时间点。

要使用所有选项的缺省值来创建一个名为 order_seq 的序列，在应用程序中或通过使用动态 SQL 语句来发出以下语句:

```
CREATE SEQUENCE order_seq
```

此序列从 1 开始，并以 1 为增量增加，且没有上限。

此示例可以表示处理从 101 开始至 200 的一组银行支票。第一个顺序应该是从 1 到 100。序列从 101 开始并以 1 为增量增加，其上限为 200。指定 NOCYCLE 以便不会产生重复的支票号。与 CACHE 参数关联的数指定了序列值的最大数目，数据库管理器预分配此数目并将它保存在内存中。

```
CREATE SEQUENCE order_seq
   START WITH 101
         INCREMENT BY 1
   MAXVALUE 200
         NOCYCLE
   CACHE 25
```

有关这些选项和其他选项的更多信息以及权限要求，请参阅 CREATE SEQUENCE 语句。

## 生成顺序值

生成顺序值是一个常见的数据库应用程序开发问题。解决该问题的最好方法是在 SQL 中使用序列和序列表达式。每个*序列*是只能由序列表达式访问的唯一已命名数据库对象。

有两个*序列表达式*: PREVIOUS VALUE 表达式和 NEXT VALUE 表达式。PREVIOUS VALUE 表达式对指定的序列返回应用程序进程中最新生成的值。与 PREVIOUS VALUE 表达式出现在同一语句中的任何 NEXT VALUE 表达式不会影响该语句中的 PREVIOUS VALUE 表达式生成的值。NEXT VALUE 序列表达式使序列值递增并返回序列的新值。

要创建序列，请发出 CREATE SEQUENCE 语句。例如，要使用缺省属性创建一个名为 id_values 的序列，请发出以下语句:

```
CREATE SEQUENCE id_values
```

要在序列的应用程序会话中生成第一个值，请发出使用 NEXT VALUE 表达式的 VALUES 语句:

```
VALUES NEXT VALUE FOR id_values

             1
-----------
                         1

            1 record(s) selected.
```

要将列值更新为序列的下一个值，请在 UPDATE 语句中包括 NEXT VALUE 表达式，如下所示:

```
UPDATE staff
  SET id = NEXT VALUE FOR id_values
  WHERE id = 350
```

要使用序列的下一个值将新行插入到表中，请在 INSERT 语句中包括 NEXT VALUE 表达式，如下所示:

```
INSERT INTO staff (id, name, dept, job)
  VALUES (NEXT VALUE FOR id_values, 'Kandil', 51, 'Mgr')
```

## 确定何时使用标识列或序列

虽然在标识列和序列之间存在相似之处，但是也存在差别。在设计数据库和应用程序时可以使用其各自的特征。

根据您的数据库设计和使用数据库的应用程序，下列特征将帮助您确定何时使用标识列以及何时使用序列。

**标识列特征**

- 标识列自动为单个表生成值。
- 当将标识列定义为 GENERATED ALWAYS 时，始终由数据库管理器生成所用的值。在修改表的内容期间，不允许应用程序来提供它们自己的值。
- 在插入行后，通过使用 IDENTITY_VAL_LOCAL() 函数或通过使用 SELECT FROM INSERT 语句从插入中重新选择标识列，可以检索生成的标识值。
- LOAD 实用程序可以生成标识值。

**序列特征**

- 未使序列与任何一个表相关。
- 序列生成可在任何 SQL 或 XQuery 语句中使用的顺序值。

由于任何应用程序可以使用序列，所以有两种表达式可用来控制如何检索指定序列中的下一个值和在正在执行的语句之前生成的值。对于当前会话中的先前语句，PREVIOUS VALUE 表达式对指定序列返回最新生成的值。NEXT VALUE 表达式对指定序列返回下一个值。使用这些表达式允许在几个表内的几个 SQL 和 XQuery 语句中使用相同值。

## 修改序列

使用 ALTER SEQUENCE 语句修改现有序列的属性。

可以修改的序列属性包括:

- 更改将来值之间的增量
- 建立新的最小值或最大值
- 更改高速缓存序号的数目
- 更改序列是否将循环
- 更改是否必须按请求顺序生成序号
- 重新启动序列

有两种任务不是序列创建的一部分。它们是:

- **RESTART**: 将序列复位为隐式或显式指定的值，该值是在创建序列时作为起始值指定的。
- **RESTART WITH <numeric-constant>**: 将序列复位为准确的数字常数值。数字常数可以是任何正数值或负数值，而且任何小数点右边不带有非零数字。

在重新启动序列或更改为 CYCLE 之后，可能会生成重复序号。ALTER SEQUENCE 语句仅影响将来的序号。

不能更改序列的数据类型。而是必须删除当前序列，然后创建新序列，指定新的数据类型。

在改变序列时，会丢失数据库管理器未使用的所有高速缓存序列值。

## 查看序列定义

使用包含 PREVIOUS VALUE 选项的 VALUES 语句来查看与序列相关的参考信息或查看序列本身。

要显示序列的当前值，请发出使用 PREVIOUS VALUE 表达式的 VALUES 语句:

```
VALUES PREVIOUS VALUE FOR id_values

            1
-----------
                        1

           1 record(s) selected.
```

可以重复检索序列的当前值，并且在发出 NEXT VALUE 表达式之前，该序列返回的值不变。在以下示例中，PREVIOUS VALUE 表达式返回值 1，直到当前连接中的 NEXT VALUE 表达式使序列的值递增为止:

```
VALUES PREVIOUS VALUE FOR id_values

             1
-----------
                         1

            1 record(s) selected.
VALUES PREVIOUS VALUE FOR id_values

             1
-----------
                         1

            1 record(s) selected.
VALUES NEXT VALUE FOR id_values

             1
-----------
                         2

            1 record(s) selected.
VALUES PREVIOUS VALUE FOR id_values

             1
-----------
                         2

            1 record(s) selected.
```

即使另一个连接在同时使用序列值也是如此。

## 删除序列

要删除序列，请使用 DROP 语句。

在删除序列时，语句的授权标识必须具有 SYSADM 或 DBADM 权限。

可以通过使用下列命令来删除特定序列:

```
DROP SEQUENCE <sequence_name>
```

其中 <sequence_name> 是要删除的序列名，它包括隐式或显式模式名以正确标识现有的序列。

不能使用 DROP SEQUENCE 语句删除系统为 IDENTITY 列创建的序列。

一旦删除序列，那么也会删除对该序列的所有特权。

## 如何编码序列的示例

编写的许多应用程序需要使用序号来跟踪发票号、客户编号以及每次需要新项时其编号就会增大 1 的其他对象。通过使用标识列，数据库管理器可以使表中的值自动递增。虽然这项技术对于单独的表来说效果不错，但它可能不是生成多个表中需要使用的唯一值的最方便方法。

*序列*对象允许您创建在程序员控制下递增并且可以在许多表中使用的值。以下示例说明了如何为客户编号创建数据类型为 INTEGER 的序号:

```
CREATE SEQUENCE customer_no AS INTEGER
```

缺省情况下，序号从 1 开始并且每次递增 1，其数据类型为 INTEGER。应用程序需要使用 NEXT VALUE 函数来获取序列中的下一个值。此函数生成序列的下一个值，然后可以将该值用于后续 SQL 语句:

```
VALUES NEXT VALUE FOR customer_no
```

程序员可以在 INSERT 语句中使用 VALUES 函数，而不是使用此函数生成下一个数字。例如，如果 Customer 表的第一列包含客户编号，那么可以按如下所示编写 INSERT 语句:

```
INSERT INTO customers VALUES
  (NEXT VALUE FOR customer_no, 'comment', ...)
```

如果需要对插入到其他表中的操作使用序号，那么可以使用 PREVIOUS VALUE 函数来检索先前生成的值。例如，如果需要将刚刚创建的客户编号用于后续发票记录，那么 SQL 应包括 PREVIOUS VALUE 函数:

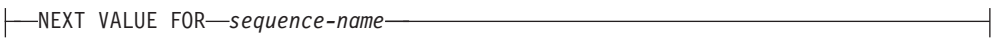
```
INSERT INTO invoices
  (34,PREVIOUS VALUE FOR customer_no, 234.44, ...)
```

PREVIOUS VALUE 函数可以在应用程序内多次使用，并且它仅返回该应用程序生成的最后一个值。后续事务可能已将序列递增至另一个值，但您看到的始终是生成的最后一个值。

## 序列引用

**sequence-reference**:

**nextval-expression**:




**prevval-expression**:

**NEXT VALUE FOR** *sequence-name*

NEXT VALUE 表达式为 *sequence-name* 指定的序列生成和返回下一个值。

**PREVIOUS VALUE FOR** *sequence-name*

对于当前应用程序进程中的先前语句，PREVIOUS VALUE 表达式为指定序列返回最新生成的值。您可以使用指定序列名称的 PREVIOUS VALUE 表达式来重复引用此值。可能会有多个 PREVIOUS VALUE 表达式的实例，指定单一语句中同一序列名，它们都返回相同的值。在分区数据库环境中，PREVIOUS VALUE 表达式可能不会返回最新生成的值。

仅在当前应用程序进程的当前或先前事务中已经引用某个指定相同序列名称的 NEXT VALUE 表达式时，才能使用 PREVIOUS VALUE 表达式（SQLSTATE 51035）。

## 注意

- 当 NEXT VALUE 表达式指定序列的名称时，将为该序列生成一个新值。但是，如果一个查询中有多个 NEXT VALUE 表达式的实例指定同一序列名，那么对于结果的每一行，序列计数器仅递增一次，且 NEXT VALUE 的所有实例对结果的某一行返回同一个值。
- 通过对第一行使用 NEXT VALUE 表达式引用序号（这会生成序列值），而对其他行使用 PREVIOUS VALUE 表达式引用序号（PREVIOUS VALUE 的实例引用当前会话中最近生成的序列值），相同序号可以在两个单独的表中用作唯一键值，如下所示:

```
INSERT INTO order(orderno, cutno)
  VALUES (NEXT VALUE FOR order_seq, 123456);

INSERT INTO line_item (orderno, partno, quantity)
  VALUES (PREVIOUS VALUE FOR order_seq, 987654, 1);
```

- 可在下列位置中指定 NEXT VALUE 和 PREVIOUS VALUE 表达式:
  - select 语句或 SELECT INTO 语句（在 select 子句中，只要该语句不包含 DISTINCT 关键字、GROUP BY 子句、ORDERBY 子句、UNION 关键字、INTERSECT 关键字或 EXCEPT 关键字）
  - INSERT 语句（在 VALUES 子句中）
  - INSERT 语句（在全查询的 select 子句中）
  - UPDATE 语句（在 SET 子句中，除了无法在 SET 子句中表达式的全查询的 select 子句中指定 NEXT VALUE 之外，它可以是被搜索或被定位的 UPDATE 语句）
  - SET Variable 语句（除了在表达式的全查询的 select 子句中之外；可以在触发器中指定 NEXT VALUE 表达式，但不能指定 PREVIOUS VALUE 表达式）
  - VALUES INTO 语句（在表达式的全查询的 select 子句中）
  - CREATE PROCEDURE 语句（在 SQL 过程的例程主体中）
  - 触发操作（triggered-action）中的 CREATE TRIGGER 语句（可以指定 NEXT VALUE 表达式，但不能指定 PREVIOUS VALUE 表达式）
- 在下列位置中不能指定 NEXT VALUE 和 PREVIOUS VALUE 表达式（SQLSTATE 428F9）:
  - 全部外连接的连接条件
  - CREATE 或 ALTER TABLE 语句中列的 DEFAULT 值
  - CREATE OR ALTER TABLE 语句中的生成列定义
  - CREATE TABLE 或 ALTER TABLE 语句中的总结表定义
  - 检查约束的条件
  - CREATE TRIGGER 语句（可以指定 NEXT VALUE 表达式，但不能指定 PREVIOUS VALUE 表达式）
  - CREATE VIEW 语句
  - CREATE METHOD 语句
  - CREATE FUNCTION 语句
  - XMLQUERY、XMLEXISTS 或 XMLTABLE 表达式的参数列表
- 此外，在下列位置中不能指定 NEXT VALUE 表达式（SQLSTATE 428F9）:
  - CASE 表达式
  - 聚集函数的参数列表

  - 非以上明确允许的上下文中的子查询
  - 外层 SELECT 为其包含了 DISTINCT 运算符的 SELECT 语句
  - 连接的连接条件
  - 外层 SELECT 为其包含了 GROUP BY 子句的 SELECT 语句
  - SELECT 语句，外层 SELECT 为该语句而与另一使用 UNION、INTERSECT 或 EXCEPT 集合运算符的 SELECT 语句组合在一起
  - 嵌套表表达式
  - 表函数的参数列表
  - 最外层的 SELECT 语句、DELETE 或 UPDATE 语句的 WHERE 子句
  - 最外层的 SELECT 语句的 ORDER BY 子句
  - 表达式的全查询的 select 子句（在 UPDATE 语句的 SET 子句中）
  - SQL 例程中的 IF、WHILE、DO...UNTIL 或 CASE 语句
- 当为序列生成值时，该值被使用，下次请求值时，将会生成新的值。即使包含 NEXT VALUE 表达式的语句失败或回滚时，情况也是如此。

  如果 INSERT 语句在该列的 VALUES 列表中包括 NEXT VALUE 表达式，并且如果在执行 INSERT 期间的某一刻发生错误（它可能是在生成下一序列值时发生的问题或与另一列的值有关的问题），那么将会发生插入故障（SQLSTATE 23505），并且为该序列生成的值被视为已使用。在某些情况下，重新发出相同的 INSERT 语句可能会导致成功。

  例如，考虑某一个因存在为其使用 NEXT VALUE 的列的唯一索引而发生的错误以及生成的序列值已存在于索引中。为该序列生成的下一个值可能是不存在于索引中的值，因此后续的 INSERT 将会成功。
- 如果在为序列生成值时超出了该序列的最大值（或者递降顺序的最小值），并且不允许循环，那么将会发生错误（SQLSTATE 23522）。在此情况下，用户可以修改序列以扩大可接受值的范围，或者为该序列启用循环，或者废弃和创建具有较大值范围的另一数据类型的新序列。

  例如，序列可能已使用 SMALLINT 数据类型进行定义，并且该序列最终用完可分配的值。废弃然后重新创建具有新定义的序列，以将该序列重新定义为 INTEGER。
- 对游标的 select 语句中的 NEXT VALUE 表达式的引用将引用为结果表行生成的值。为从数据库访存的每一行的 NEXT VALUE 表达式生成序列值。如果在客户机处进行阻塞，那么可能在处理 FETCH 语句之前已经在服务器处生成了值。当存在结果表行阻塞时，这就可能会发生。如果客户机应用程序未显式访存数据库具体化的所有行，那么应用程序将不会看到所有生成的序列值的结果（对于未返回的具体化的行来说）。
- 对游标的 select 语句中的 PREVIOUS VALUE 表达式的引用将引用在打开游标之前为指定的序列生成的值。然而，对于后续语句中的指定序列，关闭游标会影响 PREVIOUS VALUE 返回的值，甚至对于重新打开游标时的同一语句而言，情况也是如此。当游标的 select 语句包括对同一序列名的 NEXT VALUE 的引用时，就会存在此情况。
- *兼容性*
  - 为了与先前版本的 DB2 相兼容:
    - 可指定 NEXTVAL 和 PREVVAL 来代替 NEXT VALUE 和 PREVIOUS VALUE

– 为了与 IBM IDS 兼容:
  - 可指定 *sequence-name*.NEXTVAL 来代替 NEXT VALUE FOR *sequence-name*
  - 可指定 *sequence-name*.CURRVAL 来代替 PREVIOUS VALUE FOR *sequence-name*

## 示例

假设存在一个名为“order”的表以及创建了一个名为“order_seq”的序列，如下所示:

```
CREATE SEQUENCE order_seq
  START WITH 1
  INCREMENT BY 1
  NO MAXVALUE
  NO CYCLE
  CACHE 24
```

以下是如何使用 NEXT VALUE 表达式来生成“order_seq”序号的一些示例:

```
INSERT INTO order(orderno, custno)
  VALUES (NEXT VALUE FOR order_seq, 123456);
```

或者

```
UPDATE order
  SET orderno = NEXT VALUE FOR order_seq
  WHERE custno = 123456;
```

或者

```
VALUES NEXT VALUE FOR order_seq INTO :hv_seq;
```

# 第 16 章 视图

*视图*是高效率的数据呈示方法（无需维护数据）。视图不是实际的表，不需要永久存储器。“虚拟表”是即创建即使用的。

*视图*提供了另一种查看一个或多个表中的数据的方法，它是结果表的已命名规范。规范指的是每次在 SQL 语句中引用视图时运行的 SELECT 语句。视图和表一样具有列和行。可以像使用表一样将所有视图用于数据检索。是否可以在插入、更新或删除操作中使用视图取决于它的定义。

视图可以包括它所基于的表中的所有或某些列或行。例如，可以在视图中连接一个部门表和一个职员表，以便可以列示特定部门中的所有职员。

图 28 显示了表与视图之间的关系。

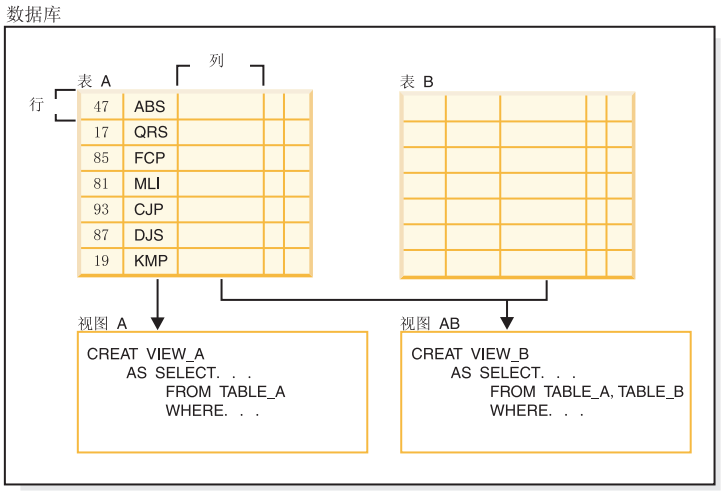



*图 28. 表与视图之间的关系*

因为视图允许多个用户查看相同数据的不同表示，所以可以使用视图来控制对敏感数据的访问。例如，几个用户正在访问关于职员的数据表。经理可以看到关于他/她的职员的数据，但看不到关于其他部门中的职员的数据。招聘专员可以看到所有职员的聘用日期，但看不到他们的薪水；财务人员可以看到薪水，但看不到聘用日期。这些用户中的每个用户都使用派生自表的视图。每个视图都显示为一个表，并且具有自己的名称。

当视图列直接派生自基本表的列时，该视图列将继承适用于该表列的所有约束。例如，如果视图包括其表的外键，那么使用该视图的插入和更新操作应遵守与该表相同的引用约束。此外，如果视图的表是一个父表，那么使用该视图的删除和更新操作应遵守与对表执行删除和更新操作相同的规则。

视图可以从结果表派生每一列的数据类型，或者使类型基于用户定义的结构化类型的属性。这种视图称为*带类型视图*。类似于类型表，带类型视图可以是视图层次结构的

© Copyright IBM Corp. 1993, 2010

一部分。*子视图*继承其*超视图*的列。术语*子视图*适用于一个带类型视图以及视图层次结构中该带类型视图下的所有带类型视图。视图 V 的*正确子视图*是在带类型视图层次结构中视图 V 下面的一个视图。

视图可能变得不可用（例如，在删除表时）；如果出现这种情况，那么该视图将不再适于 SQL 操作。

## 设计视图

*视图*提供了另一种查看一个或多个表中的数据的方法，它是结果表的已命名规范。

规范指的是每次在 SQL 语句中引用视图时运行的 SELECT 语句。视图和基本表一样具有列和行。可以像使用表一样将所有视图用于数据检索。是否可以在插入、更新或删除操作中使用视图取决于它的定义。

视图按它们允许的操作分类。它们可以是:

- 可删除
- 可更新
- 可插入
- 只读

根据视图的更新功能建立视图类型。分类指示允许对视图执行的 SQL 操作类型。

引用约束和检查约束相互独立。它们不影响视图分类。

例如，由于引用约束，您可能无法将行插入到表中。如果使用该表创建视图，那么也不能使用视图来插入该行。但是，如果该视图符合可插入视图的所有规则，那么仍将它视为可插入视图。这是因为插入限制是针对表，而不是针对视图定义。

有关更多信息，请参阅 CREATE VIEW 语句。

### 系统目录视图

数据库管理器维护一组表和视图，这些表和视图包含关于数据库管理器所控制的数据的信息。这些表和视图统称为*系统目录*。

系统目录包含关于数据库对象（例如，表、视图、索引、程序包和函数）的逻辑和物理结构的信息。它还包含统计信息。数据库管理器确保系统目录中的描述始终准确。

系统目录视图类似于任何其他数据库视图。可以使用 SQL 语句来查询系统目录视图中的数据。可以使用一组可更新的系统目录视图来修改系统目录中的某些值。

### 使用检查选项的视图

定义了 WITH CHECK OPTION 的视图将针对该视图的 SELECT 语句强制检查任何修改或插入的行。使用检查选项的视图也称为*对称视图*。例如，仅返回部门 10 中的职员的对称视图不允许插入其他部门中的职员。因此，此选项将确保数据库中修改的数据的完整性，并在 INSERT 或 UPDATE 操作期间违反条件时返回错误。

如果应用程序无法将需要的规则定义为表检查约束，或者规则不适用于数据的所有用法，那么可以使用另一种方法在应用程序逻辑中实施规则。可以考虑创建一个表视

图，其对数据的条件包括在指定的 WHERE 子句和 WITH CHECK OPTION 子句中。此视图定义将数据检索限于对应用程序有效的行集。此外，如果您可以更新该视图，那么 WITH CHECK OPTION 子句将更新、插入和删除操作限于适用于应用程序的行。

不能对下列视图指定 WITH CHECK OPTION:

- 使用只读选项定义的视图（只读视图）
- 引用 NODENUMBER 或 PARTITION 函数、非确定性函数（例如，RAND）或使用外部操作的函数的视图
- 带类型视图

## 示例 1

以下是一个使用 WITH CHECK OPTION 的视图定义的示例。需要此选项以确保始终检查条件。该视图确保 DEPT 始终为 10。这将限制 DEPT 列的输入值。使用视图来插入新值时，始终强制执行 WITH CHECK OPTION:

```
CREATE VIEW EMP_VIEW2
  (EMPNO, EMPNAME, DEPTNO, JOBTITLE, HIREDATE)
AS SELECT ID, NAME, DEPT, JOB, HIREDATE FROM EMPLOYEE
  WHERE DEPT=10
WITH CHECK OPTION;
```

如果在 INSERT 语句中使用此视图，那么当 DEPTNO 列的值不是 10 时将拒绝行。一定要记住，在未指定 WITH CHECK OPTION 的情况下，在修改期间不会进行数据验证。

如果在 SELECT 语句中使用此视图，那么将会调用条件（WHERE 子句）并且生成的表仅包含匹配的数据行。也就是说，WITH CHECK OPTION 不影响 SELECT 语句的结果。

## 例 2

使用视图，可以使表数据的子集可用于一个应用程序，并验证要插入或更新的数据。视图可以有与原始表中对应列的名称不同的列名。例如:

```
CREATE VIEW <name> (<column>, <column>, <column>)
SELECT <column_name> FROM <table_name>
      WITH CHECK OPTION
```

## 示例 3

使用视图使程序和最终用户查询可以灵活地查看表数据。

下列 SQL 语句创建 EMPLOYEE 表的视图，它列示部门 A00 的所有职员及其职员姓名和电话号码:

```
CREATE VIEW EMP_VIEW (DA00NAME, DA00NUM, PHONENO)
        AS SELECT LASTNAME, EMPNO, PHONENO FROM EMPLOYEE
        WHERE WORKDEPT = 'A00'
        WITH CHECK OPTION
```

此语句的第一行对该视图命名并定义它的列。名称 EMP_VIEW 在 SYSCAT.TABLES 中的模式内必须是唯一的。尽管不包含数据，视图名看上去仍象一个表名。该视图将有称为 DA00NAME、DA00NUM 和 PHONENO 的三列，它们与 EMPLOYEE 表中的列

LASTNAME、EMPNO 和 PHONENO 相对应。列示的列名按一一对应的关系应用于 SELECT 语句的选择列表。如果不指定列名，那么视图使用与 SELECT 语句的结果表的列相同的名称。

第二行是描述要从数据库选择哪些值的 SELECT 语句。它可以包括子句: ALL、DISTINCT、FROM、WHERE、GROUP BY 和 HAVING。为视图提供列的数据对象的一个或多个名称必须跟在 FROM 子句后面。

### 示例 4

WITH CHECK OPTION 子句指示必须根据该视图定义检查该视图的任何更新的行或插入的行，如果它不符合，那么将其拒绝。这增强了数据完整性，但是需要其他的处理。如果将此子句省略，那么不会根据视图定义检查插入和更新。

以下 SQL 语句使用 SELECT AS 子句创建基于 EMPLOYEE 表的相同视图:

```
CREATE VIEW EMP_VIEW
         SELECT LASTNAME AS DA00NAME,
                       EMPNO AS DA00NUM,
                       PHONENO
        FROM EMPLOYEE            WHERE WORKDEPT = 'A00'
         WITH CHECK OPTION
```

对于此示例，EMPLOYEE 表中可能有工资信息，它不应对每个人可用。但是，职员的电话号码通常应该是可以访问的。在这种情况下，可以仅根据 LASTNAME 和 PHONENO 列创建一个视图。可将该视图的访问权授予 PUBLIC，而将整个 EMPLOYEE 表的访问权限制在具有查看工资信息授权的那些人范围内。

## 嵌套视图定义

如果视图基于另一个视图，那么必须求值的谓词数目取决于指定的 WITH CHECK OPTION。

如果未使用 WITH CHECK OPTION 定义视图，那么在检查任何插入或更新操作的数据有效性时不会使用该视图的定义。但是，如果视图直接或间接基于另一个使用 WITH CHECK OPTION 定义的视图，那么在检查任何插入或更新操作时将使用该超视图的定义。

如果视图是使用 WITH CASCADED CHECK OPTION 或仅 WITH CHECK OPTION（CASCADED 是 WITH CHECK OPTION 的缺省值）定义的，那么在检查任何插入或更新操作时将使用该视图的定义。此外，该视图将继承它所基于的任何可更新视图的搜索条件。即使这些视图不包含 WITH CHECK OPTION，也继承这些条件。然后，继承的条件成倍在一起，以便符合对该视图或基于该视图的任何视图的任何插入或更新操作应用的约束。

例如，如果视图 V2 基于视图 V1，并且 V2 的检查选项是使用 WITH CASCADED CHECK OPTION 定义的，那么在对视图 V2 执行 INSERT 和 UPDATE 语句时，将对这两个视图的谓词进行求值:

```
CREATE VIEW EMP_VIEW2 AS
  SELECT EMPNO, EMPNAME, DEPTNO FROM EMP
    WHERE DEPTNO = 10
WITH CHECK OPTION;
```

以下示例显示使用 WITH CASCADED CHECK OPTION 的 CREATE VIEW 语句。视图 EMP_VIEW3 是在视图 EMP_VIEW2 的基础上创建的，而 EMP_VIEW2 是使用

WITH CHECK OPTION 创建的。如果要在 EMP_VIEW3 中插入或更新记录，那么该记录的值应该为 DEPTNO=10 和 EMPNO=20。

```
CREATE VIEW EMP_VIEW3 AS
  SELECT EMPNO, EMPNAME, DEPTNO FROM EMP_VIEW2
    WHERE EMPNO > 20
WITH CASCADED CHECK OPTION;
```

**注**: 即使 EMP_VIEW2 不包含 WITH CHECK OPTION，在对 EMP_VIEW3 执行插入或更新操作时也会强制执行条件 DEPTNO=10。

在创建视图时，还可以指定 WITH LOCAL CHECK OPTION。如果视图是使用 LOCAL CHECK OPTION 定义的，那么在检查任何插入或更新操作时将使用该视图的定义。但是，该视图不继承它所基于的任何可更新视图的搜索条件。

## 可删除视图

根据定义视图的方式，视图可以是可删除视图。可删除视图是可以对其成功发出 DELETE 语句的视图。

只有在遵循下列规则的情况下，一个视图才能被视为可删除视图:

- 外部全查询的每个 FROM 子句仅标识一个表（不带有 OUTER 子句）、可删除视图（不带有 OUTER 子句）、可删除的嵌套表表达式或可删除的公共表表达式。
- 数据库管理器应该能够使用视图定义来派生表中要删除的行。某些操作使得这不可能实现:
  - 使用 GROUP BY 子句或列函数将多行分组为一行将导致原始行丢失并且使得视图不可删除。
  - 同样，从 VALUES 派生行时，没有要删除行的表。视图也将不可删除。
- 外部全查询不使用 GROUP BY 或 HAVING 子句。
- 外部全查询的选择列表中不包括列函数。
- 外部全查询不使用集操作（UNION、EXCEPT 或 INTERSECT），但 UNION ALL 除外。
- UNION ALL 的操作数中的表不能是相同的表，并且每个操作数必须可删除。
- 外部全查询的选择列表不包括 DISTINCT。

一个视图必须符合上面列示的所有规则才能被视为可删除视图。例如，下列视图是可删除视图。它遵循可删除视图的所有规则。

```
CREATE VIEW deletable_view
  (number, date, start, end)
AS      SELECT number, date, start, end
  FROM employee.summary
  WHERE date='01012007'
```

## 可插入视图

可插入视图允许您使用视图定义来插入行。如果对视图定义了用于插入操作的 INSTEAD OF 触发器，或者视图中的至少一列可更新（与用于更新的 INSTEAD OF 触发器无关）并且视图的全查询不包括 UNION ALL，那么该视图是可插入视图。当且仅当给定行满足正好一个基础表的检查约束时，才能将该行插入到视图中（包括 UNION ALL）。要插入到包含不可更新列的视图中，必须从列列表中省略这些列。

下面显示的视图是一个可插入视图。但是，在本示例中，尝试插入视图将失败。这是因为表中存在不接受空值的列。这些列中的某些列未出现在视图定义中。尝试使用视图来插入值时，数据库管理器会尝试将一个空值插入到 NOT NULL 列中。不允许执行此操作。

```
CREATE VIEW insertable_view
  (number, name, quantity)
AS    SELECT number, name, quantify FROM ace.supplies
```

**注:** 对表定义的约束与可以使用基于该表的视图执行的操作无关。

## 可更新视图

可更新视图是一种特殊的可删除视图。如果可删除视图中的至少一列可更新，那么该可删除视图就变成可更新视图。

当满足下列所有规则时，视图中的一列将可更新:

- 视图是可删除视图。
- 列解析为表列（不使用解引用操作）并且未指定 READ ONLY 选项。
- 如果视图的全查询包含 UNION ALL，那么 UNION ALL 的操作数的所有相应列具有完全匹配的数据类型（包括长度或精度和小数位）以及完全匹配的缺省值。

以下示例使用无法更新的常量值。但是，视图是可删除视图并且该视图中至少一列可更新。因此，它是可更新视图。

```
CREATE VIEW updatable_view
  (number, current_date, current_time, temperature)
AS      SELECT number, CURRENT DATE, CURRENT TIME, temperature)
FROM weather.forecast
WHERE number = 300
```

## 只读视图

如果一个视图不可删除、更新或插入，那么该视图是*只读*视图。如果一个视图不符合可删除视图的至少一个规则，那么该视图是只读视图。

SYSCAT.VIEWS 目录视图中的 READONLY 列指示视图是只读（R）视图。

下面显示的示例不是可删除视图，因为它使用了 DISTINCT 子句并且 SQL 语句涉及多个表:

```
CREATE VIEW read_only_view
  (name, phone, address)
AS    SELECT DISTINCT viewname, viewphone, viewaddress
FROM employee.history adam, employer.dept sales
WHERE adam.id = sales.id
```

# 创建视图

视图可从一个或多个表、昵称或视图中派生，且可以在检索数据时与表互换使用。当对视图中显示的数据进行更改时，该数据会在表中自行更改。在创建视图之前，视图所基于的表、昵称或视图必须已经存在。

可以创建视图来限制对敏感数据的访问，同时又允许对其他数据进行更多的一般访问。

当插入到一个视图中，而其视图定义中的选择列表直接或间接地包括一个表的标识列的名称时，同一规则适用，就像 INSERT 语句直接引用该表的标识列一样。

除按上述方式使用视图外，视图还可以用于:

- 改变表而不影响应用程序。这可通过创建一个基于基础表的视图来完成。使用基础表的应用程序不会因新视图的创建而受影响。新的应用程序可将创建的视图用于与那些使用基础表的应用程序不同的目的。
- 对一列中的值求和，选择最大值，或计算平均值。
- 访问一个或多个数据源中的信息。可在 CREATE VIEW 语句内引用昵称，并可创建多个位置/全局视图（该视图可以连接位于不同系统中多个数据源的信息）。

  当使用标准的 CREATE VIEW 语法创建一个引用昵称的视图时，将看到一个警告，它警告您视图用户的认证标识而不是视图创建者的认证标识将用于访问数据源处的基本对象。使用 FEDERATED 关键字阻止此警告。

带类型视图以预定义的结构化类型为基础。可使用 CREATE VIEW 语句来创建带类型视图。

创建视图的一个替代方法是使用嵌套的或公共表表达式，以减少目录查找，并提高性能。

下面显示了样本 CREATE VIEW 语句。基础表 EMPLOYEE 具有 SALARY 列和 COMM 列。为了安全起见，仅根据 ID、NAME、DEPT、JOB 和 HIREDATE 列创建此视图。此外，对 DEPT 列的访问受限制。此定义仅显示属于 DEPTNO 为 10 的部门的职员信息。

```
CREATE VIEW EMP_VIEW1
  (EMPID, EMPNAME, DEPTNO, JOBTITLE, HIREDATE)
  AS SELECT ID, NAME, DEPT, JOB, HIREDATE FROM EMPLOYEE
    WHERE DEPT=10;
```

在定义视图后，可以指定访问特权。由于只能访问表的受限视图，所以指定访问特权可以提供数据安全性。如上所示，视图可以包含 WHERE 子句以限制对某些行的访问，或者可以包含列子集以限制对某些数据列的访问。

视图中的列名不必与基本表的列名匹配。表名具有关联的模式，视图名也一样。

定义视图后，就可以在诸如 SELECT、INSERT、UPDATE 和 DELETE 之类的语句中使用它（但具有一些限制）。DBA 可以决定提供一组用户，他们对视图具有的特权级别比对表的特权级别要高。

## 创建使用用户定义的函数（UDF）的视图

在创建使用 UDF 的视图后，只要该视图存在，它就始终使用这个 UDF，即使在稍后创建了其他具有相同名称的 UDF 也是如此。如果需要挑选新的 UDF，那么需要重新创建视图。

下列 SQL 语句创建一个在其定义中带有函数的视图:

```
CREATE VIEW EMPLOYEE_PENSION (NAME, PENSION)
    AS SELECT NAME, PENSION(HIREDATE,BIRTHDATE,SALARY,BONUS)
    FROM EMPLOYEE
```

UDF 函数 PENSION 根据涉及 HIREDATE、BIRTHDATE、SALARY 和 BONUS 的一个公式来计算一个职员应当得到的当前退休金。

## 修改带类型视图

可以更改带类型视图的某些属性，而不需要删除并重新创建该视图。一个这种属性就是将作用域添加至带类型视图的引用列。

ALTER VIEW 语句通过改变引用类型列来添加作用域，以修改现有带类型视图定义。DROP 语句删除带类型视图。您还可以:

- 通过 INSTEAD OF 触发器修改带类型视图的内容
- 改变带类型视图以启用统计信息收集

对带类型视图的基本内容所作的更改需要使用触发器。对带类型视图的其他更改需要删除带类型视图然后重新创建该视图。

在 ALTER VIEW 语句中列名的数据类型必须是 REF（类型表名或带类型视图名的类型）。

虽然程序包和高速缓存的动态语句都被标记为无效，但是不会影响其他数据库对象，如表和索引。

要使用命令行来改变带类型视图，请输入:

```
ALTER VIEW <view_name> ALTER <column_name>
         ADD SCOPE <typed table or view name>
```

## 恢复不可用视图

不可用视图是不再能用于 SQL 语句的视图。

视图可能变得*不可用*:

- 由于撤销了对基础表的特权
- 在删除了表、别名或函数的情况下。
- 在超视图变得不可用的情况下。（超视图是另一个带类型视图或子视图所基于的带类型视图。）
- 当删除它们所从属的视图时。

下列步骤可以帮助您恢复不可用视图:

1. 确定最初用于创建该视图的 SQL 语句。可以从 SYSCAT.VIEW 目录视图的 TEXT 列获取此信息。
2. 将当前模式设置为 QUALIFIER 列的内容。
3. 将函数路径设置为 FUNC_PATH 列的内容。
4. 使用 CREATE VIEW 语句并使用相同的视图名和相同的定义来重新创建该视图。
5. 使用 GRANT 语句重新授予先前在该视图上授予的所有特权。（注意，在不可用视图上授予的所有特权都被撤销。）

如果不希望恢复不可用视图，可以使用 DROP VIEW 语句显式删除它，或者可以使用相同的名称和不同的定义来创建一个新视图。

不可用视图只在 SYSCAT.TABLES 和 SYSCAT.VIEWS 目录视图中有条目；在 SYSCAT.TABDEP、SYSCAT.TABAUTH、SYSCAT.COLUMNS 和 SYSCAT.COLAUTH 目录视图中的所有条目已被除去。

## 删除视图

使用 DROP VIEW 语句来删除视图。从属于要删除的视图的任何其他视图将变得不可用。

要使用命令行来删除视图，请输入:

```
DROP VIEW <view_name>
```

以下示例显示如何删除名为 EMP_VIEW 的视图:

```
DROP VIEW EMP_VIEW
```

对于表层次结构，可以在一条语句中删除整个视图层次结构，方法是命名该层次结构的根视图，如以下示例所示:

```
DROP VIEW HIERARCHY VPerson
```

# 第 4 部分 参考

© Copyright IBM Corp. 1993, 2010

# 第 17 章 符合命名规则

## 命名规则

为所有对象、用户名、密码和组命名时都要遵循一些规则。某些规则与所用的平台有关。

例如，有的规则是关于名称中的大小写使用。

- 在 UNIX 平台上，名称必须小写。
- 在 Windows 平台上，名称可以是大写，小写或大小写混合。

除非另有指定，否则所有名称均可以包括下列字符:

- A 到 Z。当在大多数名称中使用时，字符 A 至 Z 将从小写形式转换为大写形式。
- 0 至 9。
- ! % ( ) { } . - ^ ~ _（下划线） @、#、$ 和空格。
- \（反斜杠）。

名称不能以数字或下划线字符开始。

不要使用 SQL 保留字来命名表、视图、列、索引或授权标识。

有一些其他特殊字符，它们所起的作用取决于操作系统以及使用 DB2 数据库的位置。虽然它们可能生效，但并不保证它们会生效。建议在数据库中命名对象时不要使用这些其他特殊字符。

用户名和组名同样需要遵循相关系统对特定操作系统强制实施的规则。例如，在 Linux 和 UNIX 平台上，允许用户名和主组名中的字符必须是小写的 a 到 z、0 到 9 以及下划线 _（如果用户名和主组名不是以 0 到 9 开头）。

长度必须小于或等于*SQL Reference*中的『SQL 和 XML 限制』中列示的长度。

还需要考虑对象命名规则、NLS 环境中的命名规则以及 Unicode 环境中的命名规则。

**对 AUTHID 标识的限制**: DB2 数据库系统的版本 9.5 和更高版本允许您使用 128 个字节的授权标识，但是，当授权标识被解释为操作系统用户标识或组名时，应遵循操作系统命名限制（例如，Linux 和 UNIX 操作系统将用户标识和组名的长度限制为 8 个字符，Windows 操作系统将用户标识和组名的长度限制为 30 个字符）。因此，虽然您可以授予一个 128 字节的授权标识，但是作为一个具有该授权标识的用户，您却无法进行连接。如果您编写自己的安全插件，那么应该能够充分利用授权标识的扩展大小。例如，您可以为安全插件指定一个 30 字节的用户标识，并且在认证期间它可能会返回一个 128 字节的授权标识，您可以使用此授权标识进行连接。

© Copyright IBM Corp. 1993, 2010

## DB2 对象命名规则

所有对象都遵从“一般命名规则”。另外，某些对象具有下表所示的其他限制。

*表 51. 数据库、数据库别名和实例命名规则*

| 对象 | 准则 |
| --- | --- |
| • 数据库<br>• 数据库别名<br>• 实例 | • 在编目数据库名称的位置中，这些数据库名称必须是唯一的。在 Linux 和 UNIX 实现上，此位置是目录路径，而在 Windows 实现上，它是逻辑磁盘。<br>• 在系统数据库目录中，数据库别名必须是唯一的。创建新数据库时，别名缺省为数据库名称。因此，就不能使用作为数据库别名存在的名称来创建数据库，即使不存在使用该名称的数据库。<br>• 数据库、数据库别名和实例名长度必须小于或等于 8 个字节<br>• 在 Windows 上，任何实例都不能与服务名称同名。<br>**注**：为避免潜在的问题，在想要在通信环境中使用数据库的情况下，不要在数据库名称中使用特殊字符 @、# 和 $。而且，因为并非所有键盘都使用这些字符，所以，如果打算使用另一种语言版本的数据库，不要使用这些特殊字符。 |

*表 52. 数据库对象命名规则*

| 对象 | 准则 |
| --- | --- |
| • 别名<br>• 审计策略<br>• 缓冲池<br>• 列<br>• 事件监视器<br>• 索引<br>• 方法<br>• 节点组<br>• 程序包<br>• 程序包版本<br>• 角色<br>• 模式<br>• 存储过程<br>• 表<br>• 表空间<br>• 触发器<br>• 可信上下文<br>• UDF<br>• UDT<br>• 视图 | 这些对象的长度必须小于或等于以下位置所列示的长度：SQL 和 XML 限制。对象名还可以包括：<br>• 有效的重音字符（例如，ö）<br>• 除了多字节空格之外的多字节字符（对于多字节环境）<br>程序包名称和程序包版本还可以包括句号（.）、连字符（-）和冒号（:） |

*表 53. 联合数据库对象命名规则*

| 对象 | 准则 |
| --- | --- |
| • 函数映射<br>• 索引规范<br>• 昵称<br>• 服务器<br>• 类型映射<br>• 用户映射<br>• 包装器 | 这些对象的长度必须小于或等于以下位置所列示的长度: SQL 和 XML 限制。联合数据库对象的名称还可以包括:<br>• 有效的强调字母（例如 ö）<br>• 除了多字节空格之外的多字节字符（对于多字节环境） |

### 定界标识和对象名

可以使用关键字。如果在某个上下文中使用了一个关键字，而该关键字还可以解释为 SQL 关键字，那么必须将其指定为定界标识。

使用定界标识时，可能会创建违反这些命名规则的对象；但是，如果随后使用该对象，那么可能导致错误。例如，如果您创建一列，其名称中包括 + 号或 - 号，然后又在索引中使用该列，那么当您试图重组该表时将遇到问题。

### 其他模式名信息

- 用户定义的类型（UDT）不能具有长度超过以下位置所列示的长度的模式名: SQL 和 XML 限制。
- 下列模式名是保留字，不能使用: SYSCAT、SYSFUN、SYSIBM 和 SYSSTAT。
- 要避免将来的潜在迁移问题，切勿使用以 SYS 开头的模式名。数据库管理器不允许您使用以 SYS 开头的模式名来创建触发器、用户定义的类型或用户定义的函数。
- 建议不要将 SESSION 用作模式名。已声明临时表必须用 SESSION 来限定。因此，可以让应用程序使用与持久表完全相同的名称来声明临时表，在这种情况下，应用程序逻辑可能会变得过分复杂。除非在处理已声明临时表，否则应避免使用模式 SESSION。

## 定界标识和对象名

可以使用关键字。如果在某个上下文中使用了一个关键字，而该关键字还可以解释为 SQL 关键字，那么必须将其指定为定界标识。

使用定界标识时，可能会创建违反这些命名规则的对象；但是，如果随后使用该对象，那么可能导致错误。例如，如果您创建一列，其名称中包括 + 号或 - 号，然后又在索引中使用该列，那么当您试图重组该表时将遇到问题。

## 用户、用户标识和组命名规则

用户、用户标识和组名必须遵循命名准则。

*表 54. 用户、用户标识和组命名规则*

| 对象 | 准则 |
| --- | --- |
| • 组名<br>• 用户名<br>• 用户标识 | • 组名必须小于或等于 SQL 和 XML 限制中所列示的组名长度。<br>• 在 Linux 和 UNIX 操作系统上，用户标识最多可以包含 8 个字符。<br>• 在 Windows 上，用户名最多可以包含 30 个字符。<br>• 在不使用“客户机”认证时，如果显式指定了用户名和密码，那么支持使用长度超过 SQL 和 XML 限制中所列示的用户名长度的用户名将 32 位非 Windows 客户机连接至 Windows。<br>• 名称和标识不能:<br>– 是 USERS、ADMINS、GUESTS、PUBLIC、LOCAL 或任何 SQL 保留字。<br>– 以 IBM、SQL 或 SYS 开头。 |

**注:**

1. 一些操作系统允许区分大小写的用户标识和密码。应该检查操作系统文档以了解是否是这种情况。
2. 从成功的 CONNECT 或 ATTACH 返回的授权标识被截断为 SQL 和 XML 限制中所列示的权限名长度。将省略号（...）追加至授权标识并且 SQLWARN 字段包含警告以指示截断。
3. 从用户标识和密码中除去尾部空格。

## NLS 环境中的命名规则

可以在数据库名称中使用的基本字符集包含单字节大写和小写拉丁字母（A...Z 和 a...z）、阿拉伯数字（0...9）和下划线字符（_）。

此列表增加了三个特殊字符（#、 @ 和 $）以提供与主机数据库产品的兼容性。在 NLS 环境中使用特殊字符 #、 @ 和 $ 应小心，因为它们未包括在 NLS 主机（EBCDIC）不变量字符集中。视正在使用的代码页而定，还可以使用来自扩展字符集的字符。如果要在一个多代码页环境中使用数据库，那么必须确保所有代码页都支持您计划使用的扩展字符集中的任何元素。

当命名数据库对象（如表和视图）、程序标号、主变量、游标时，也可使用扩展字符集（例如，带有区分标记的字母）中的元素。哪些字符正好可用取决于正在使用的代码页。

**DBCS 标识的扩展字符集定义**: 在 DBCS 环境中，扩展字符集包含基本字符集中的所有字符以及下列各项:

- 除双字节空间外，每个 DBCS 代码页中的所有双字节字符都是有效的字母。

- 双字节空间是特殊字符。
- 在每个混合的代码页中可用的单字节字符被分配给各种类别，如下所示:

| 类别 | 在每个混合代码页内有效的代码点 |
| --- | --- |
| 数字 | x30-39 |
| 字母 | x23-24、x40-5A、x61-7A 和 xA6-DF（A6-DF 仅用于代码页 932 和 942） |
| 特殊字符 | 所有其他有效的单字节字符代码点 |

## Unicode 环境中的命名规则

在 Unicode 数据库中，所有标识都使用多字节 UTF-8。因此，对于 DB2 数据库系统允许在其中使用扩展字符集中的字符（例如，重音字符或多字节字符）的标识，可以在该标识中使用任何 UCS-2 字符。

客户机可输入其环境支持的任何字符，并且标识中的所有字符都由数据库管理器转换为 UTF-8。在为 Unicode 数据库指定标识中的本地语言字符时，必须考虑以下两点:

- 每个非 ASCII 字符均需要两个到四个字节。因此，一个 *n* 字节标识只能容纳 *n*/4 至 *n* 个字符，这取决于 ASCII 与非 ASCII 字符的比率。如果仅有一个或两个非 ASCII 字符（例如，强调字符），那么该限制接近 *n* 个字符，而对于全部由非 ASCII 字符组成的标识（例如，日语），只能使用 *n*/4 至 *n*/3 个字符。
- 如果要在不同的客户机环境中输入标识，那么应该使用这些客户机可用的公共字符子集来定义标识。例如，如果要从拉丁语系 1、阿拉伯语和日语环境中访问 Unicode 数据库，那么所有标识应该严格限制为 ASCII。

# 第 18 章 SQL 和 XML 限制

下表描述了某些 SQL 和 XML 限制。遵循限制最多的情况可以帮助您设计易于移植的应用程序。

表 55 列示字节方面的限制。当创建标识时，从应用程序代码页转换为数据库代码页之后，将强制执行这些限制。在从数据库中检索标识时，从数据库代码页转换为应用程序代码页之后，也会强制执行这些限制。如果在这些进程期间超出标识长度限制，那么将会发生截断或返回错误。

字符限制会因数据库的代码页和应用程序的代码页不同而异。例如，因为 UTF-8 字符的宽度可以从 1 至 4 字节，所以限制为 128 个字节的 Unicode 表中的标识的字符限制将从 32 至 128 个字符，具体情况视所使用的字符而定。如果在转换为数据库代码页之后试图创建其名称长度超过此表的限制的标识，那么将会返回错误。

存储标识名称的应用程序必须能够在转换代码页之后处理潜在增长的标识大小。当从目录中检索标识时，标识将被转换为应用程序代码页。从数据库代码页转换为应用程序代码页可能导致标识长度超过表的字节限制。如果应用程序声明的主变量在代码页转换后无法存储整个标识，那么它会被截断。如果这种情况不可接受，那么可以增大主变量大小，以便能够接受整个标识名称。

相同的规则适用于 DB2 实用程序检索数据并将其转换为用户指定的代码页。如果“导出”等 DB2 实用程序正在检索数据和强制转换为用户指定的代码页（使用导出 CODEPAGE 修饰符或 **DB2CODEPAGE** 注册表变量），并且标识因代码页转换而超出了此表所描述的限制，那么可能会返回错误或截断标识。

*表 55. 标识长度限制*

| 描述 | 最大值（以字节计） |
|---|---|
| 别名 | 128 |
| 属性名称 | 128 |
| 审计策略名称 | 128 |
| 权限名（只能是单字节字符） | 128 |
| 缓冲池名称 | 18 |
| 列名[2] | 128 |
| 约束名 | 128 |
| 相关名 | 128 |
| 游标名 | 128 |
| 数据分区名 | 128 |
| 数据源列名 | 255 |
| 数据源索引名 | 128 |
| 数据源名 | 128 |
| 数据源表名（*remote-table-name*） | 128 |
| 数据库分区组名 | 128 |
| 数据库分区名 | 128 |

© Copyright IBM Corp. 1993, 2010

*表 55. 标识长度限制 (续)*

| 描述 | 最大值（以字节计） |
|---|---|
| 事件监视器名称 | 128 |
| 外部程序名 | 128 |
| 函数映射名 | 128 |
| 组名 | 128 |
| 主机标识[1] | 255 |
| 数据源用户的标识（*remote-authorization-name*） | 128 |
| SQL 过程中的标识（对于循环标识、标号、结果集定位器、语句名称和变量名，这指的是条件名） | 128 |
| 索引名 | 128 |
| 索引扩展名 | 18 |
| 索引规范名 | 128 |
| 标号名称 | 128 |
| 名称空间统一资源标识（URI） | 1000 |
| 昵称 | 128 |
| 程序包名称 | 128 |
| 程序包版本标识 | 64 |
| 参数名 | 128 |
| 用于访问数据源的密码 | 32 |
| 过程名称 | 128 |
| 角色名 | 128 |
| 保存点名称 | 128 |
| 模式名[2] | 128 |
| 安全标号组件名称 | 128 |
| 安全标号名称 | 128 |
| 安全策略名称 | 128 |
| 序列名 | 128 |
| 服务器（数据库别名）名称 | 8 |
| 特定名称 | 128 |
| SQL 条件名 | 128 |
| SQL 变量名 | 128 |
| 语句名称 | 128 |
| 表名 | 128 |
| 表空间名 | 18 |
| 变换组名 | 18 |
| 触发器名称 | 128 |
| 可信上下文名称 | 128 |
| 类型映射名 | 18 |
| 用户定义的函数名 | 128 |
| 用户定义的方法名称 | 128 |
| 用户定义的类型名[2] | 128 |

*表 55. 标识长度限制 (续)*

| 描述 | 最大值（以字节计） |
|---|---|
| 视图名 | 128 |
| 包装器名称 | 128 |
| XML 元素名称、属性名称或前缀名称 | 1000 |
| XML 模式位置统一资源标识（URI） | 1000 |
| **注:**<br>1. 各个主语言编译器可能对变量名具有更严格的限制。<br>2. 对于用户定义的类型，SQLDA 结构被限制为存储 30 个字节的列名、18 个字节的用户定义的类型名和 8 个字节的模式名。因为在 DESCRIBE 语句中使用 SQLDA，所以使用 DESCRIBE 语句来检索列或用户定义的类型名信息的嵌入式 SQL 应用程序必须符合这些限制。 | |

*表 56. 数字限制*

| 描述 | 限制 |
|---|---|
| 最小的 SMALLINT 值 | -32 768 |
| 最大的 SMALLINT 值 | +32 767 |
| 最小的 INTEGER 值 | -2 147 483 648 |
| 最大的 INTEGER 值 | +2 147 483 647 |
| 最小的 BIGINT 值 | -9 223 372 036 854 775 808 |
| 最大的 BIGINT 值 | +9 223 372 036 854 775 807 |
| 最大的十进制精度 | 31 |
| REAL 值的最大指数（$E_{max}$） | 38 |
| 最小的 REAL 值 | -3.402E+38 |
| 最大的 REAL 值 | +3.402E+38 |
| REAL 值的最小指数（$E_{min}$） | -37 |
| 最小的正 REAL 值 | +1.175E-37 |
| 最大的负 REAL 值 | -1.175E-37 |
| DOUBLE 值的最大指数（$E_{max}$） | 308 |
| 最小的 DOUBLE 值 | -1.79769E+308 |
| 最大的 DOUBLE 值 | +1.79769E+308 |
| DOUBLE 值的最小指数（$E_{min}$） | -307 |
| 最小的正 DOUBLE 值 | +2.225E-307 |
| 最大的负 DOUBLE 值 | -2.225E-307 |
| DECFLOAT(16) 值的最大指数（$E_{max}$） | 384 |
| 最小的 DECFLOAT(16) 值[1] | -9.999999999999999E+384 |
| 最大的 DECFLOAT(16) 值 | 9.999999999999999E+384 |
| DECFLOAT(16) 值的最小指数（$E_{min}$） | -383 |
| 最小的正 DECFLOAT(16) 值 | 1.000000000000000E-383 |
| 最大的负 DECFLOAT(16) 值 | -1.000000000000000E-383 |

*表 56. 数字限制 (续)*

| 描述 | 限制 |
|---|---|
| DECFLOAT(34) 值的最大指数（$E_{max}$） | 6144 |
| 最小的 DECFLOAT(34) 值[1] | -9.999999999999999999999999999999999E+6144 |
| 最大的 DECFLOAT(34) 值 | 9.999999999999999999999999999999999E+6144 |
| DECFLOAT(34) 值的最小指数（$E_{min}$） | -6143 |
| 最小的正 DECFLOAT(34) 值 | 1.000000000000000000000000000000000E-6143 |
| 最大的负 DECFLOAT(34) 值 | -1.000000000000000000000000000000000E-6143 |

**注:**

1. 存在标准十进制浮点数的限制。有效的十进制浮点值包括特殊值 NAN、-NAN、SNAN、-SNAN、INFINITY 和 -INFINITY。此外，有效值包括次正规数。

   次正规数是其调整的指数小于 $E_{min}$ 的非零数字。对于次正规数，指数的最小值为 $E_{min}$ - (*precision*-1)，它被称为 $E_{tiny}$，其中 *precision* 是工作精度（16 或 34）。即，次正规数按 15 或 33 次方分别为 DECFLOAT(16) 或 DECFLOAT(34) 扩大接近于零的数字范围。次正规数与正规数不同，因为次正规数的最大位数小于工作精度（16 或 34）。十进制浮点不能表示具有与表示正规数那样相同精度的次正规数。DECFLOAT(34) 的最小正值次正规数为 $1x10^{-6176}$，它只包含一位数，然而 DECFLOAT(34) 的最小正值正规数为 $1.000000000000000000000000000000000x10^{-6143}$，它包含 34 位数。DECFLOAT(16) 的最小正值次正规数为 $1x10^{-398}$。

*表 57. 字符串限制*

| 描述 | 限制 |
|---|---|
| CHAR 的最大长度（以字节计） | 254 |
| VARCHAR 的最大长度（以字节计） | 32 672 |
| LONG VARCHAR 的最大长度（以字节计）[1] | 32 700 |
| CLOB 的最大长度（以字节计） | 2 147 483 647 |
| 序列化 XML 的最大长度（以字节计） | 2 147 483 647 |
| GRAPHIC 的最大长度（以双字节字符计） | 127 |
| VARGRAPHIC 的最大长度（以双字节字符计） | 16 336 |
| LONG VARGRAPHIC 的最大长度（以双字节字符计）[1] | 16 350 |
| DBCLOB 的最大长度（以双字节字符计） | 1 073 741 823 |
| BLOB 的最大长度（以字节计） | 2 147 483 647 |
| 字符常量的最大长度 | 32 672 |
| 图形常量的最大长度 | 16 336 |
| 并置字符串的最大长度 | 2 147 483 647 |
| 并置图形字符串的最大长度 | 1 073 741 823 |
| 并置二进制字符串的最大长度 | 2 147 483 647 |
| 十六进制常数的最大位数 | 32 672 |
| 运行时结构化类型列对象的最大实例（以千兆字节计） | 1 |
| 目录注释的最大大小（以字节计） | 254 |

*表 57. 字符串限制 (续)*

| 描述 | 限制 |
| --- | --- |
| **注:** | |
| 1. 不推荐（不建议）使用 LONG VARCHAR 和 LONG VARGRAPHIC 数据类型，将来的发行版中可能会将其除去。 | |

*表 58. XML 限制*

| 描述 | 限制 |
| --- | --- |
| XML 文档的最大深度（以级别计） | 125 |
| XML 模式文档的最大大小（以字节计） | 31 457 280 |

*表 59. 日期/时间限制*

| 描述 | 限制 |
| --- | --- |
| 最小的 DATE 值 | 0001-01-01 |
| 最大的 DATE 值 | 9999-12-31 |
| 最小的 TIME 值 | 00:00:00 |
| 最大的 TIME 值 | 24:00:00 |
| 最小的 TIMESTAMP 值 | 0001-01-01-00.00.00.000000 |
| 最大的 TIMESTAMP 值 | 9999-12-31-24.00.00.000000 |

*表 60. 数据库管理器限制*

| 描述 | 限制 |
| --- | --- |
| **应用程序** | |
| 预编译的程序中主变量声明的最大数目[3] | 存储器 |
| 主变量值的最大长度（以字节计） | 2 147 483 647 |
| 程序中已声明游标的最大数目 | 存储器 |
| 工作单元中已更改的行的最大数目 | 存储器 |
| 同时打开的游标的最大数目 | 存储器 |
| DB2 客户机中每个进程的最大连接数 | 512 |
| 事务中同时打开的 LOB 定位器的最大数目 | 4 194 304 |
| SQLDA 的最大大小（以字节计） | 存储器 |
| 预编译语句的最大数目 | 存储器 |
| **缓冲池** | |
| 32 位发行版的缓冲池中的最大 NPAGES | 1 048 576 |
| 64 位发行版的缓冲池中的最大 NPAGES | 2 147 483 647 |
| 所有缓冲池槽的最大总大小（4K） | 2 147 483 646 |
| **并行性** | |
| 服务器的并发用户的最大数目[4] | 64 000 |
| 每个实例的并发用户的最大数目 | 64 000 |
| 每个数据库的并发应用程序的最大数目 | 60 000 |
| 每个实例同时使用的数据库的最大数目 | 256 |
| **约束** | |

*表 60. 数据库管理器限制 (续)*

| 描述 | 限制 |
|---|---|
| 表中的最大约束数 | 存储器 |
| 唯一约束中的最大列数（通过唯一索引受支持） | 64 |
| 唯一约束中的最大列组合长度（通过唯一索引受支持，以字节计）[9] | 8192 |
| 外键中的最大引用列数 | 64 |
| 外键中引用列的最大组合长度（以字节计）[9] | 8192 |
| 检查约束规范的最大长度（以字节计） | 65 535 |
| **数据库** | |
| 最大数据库分区号 | 999 |
| **索引** | |
| 表中的最大索引数 | 32 767 或存储器 |
| 索引键中的最大列数 | 64 |
| 索引键的最大长度（包括所有开销）[7] [9] | *indexpagesize*/4 |
| 可变索引键部件的最大长度（以字节计）[8] | 1022 或存储器 |
| SMS 表空间中的每个数据库分区的最大索引大小（以千兆字节计）[7] | 16 384 |
| 常规 DMS 表空间中的每个数据库分区的最大索引大小（以千兆字节计）[7] | 512 |
| 大型 DMS 表空间中的每个数据库分区的最大索引大小（以千兆字节计）[7] | 16 384 |
| XML 数据索引 的可变索引键部件的最大长度（以字节计）[7] | *pagesize*/4 - 207 |
| **日志记录** | |
| 最大日志序号 | 0xFFFF 0000 0000 |
| **监视** | |
| 同时处于活动状态的事件监视器的最大数目 | 32 |
| **例程** | |
| 带 LANGUAGE SQL 的过程中的最大参数数 | 32 767 |
| 带 PROGRAM TYPE MAIN 的外部过程中的最大参数数 | 32 767 |
| 带 PROGRAM TYPE SUB 的外部过程中的最大参数数 | 90 |
| 用户定义的函数中的最大参数数 | 90 |
| 例程的最大嵌套级别数 | 64 |
| SQL 路径中的最大模式数目 | 64 |
| SQL 路径的最大长度（以字节计） | 2048 |
| **安全性** | |
| 类型集或树的安全标号组件中的最大元素数 | 64 |
| 类型数组的安全标号组件中的最大元素数 | 65 535 |
| 安全策略中的安全标号组件的最大数目 | 16 |
| **SQL** | |
| SQL 语句的最大总长度（以字节计） | 2 097 152 |
| SQL 语句或视图中引用的最大表数 | 存储器 |

*表 60. 数据库管理器限制 (续)*

| 描述 | 限制 |
|---|---|
| SQL 语句中引用的最大主变量数 | 32 767 |
| 语句中的最大常量数 | 存储器 |
| 选择列表中的最大元素数[7] | 1012 |
| WHERE 或 HAVING 子句中的最大谓词数 | 存储器 |
| GROUP BY 子句中的最大列数[7] | 1012 |
| GROUP BY 子句中的列的最大总长度（以字节计）[7] | 32 677 |
| ORDER BY 子句中的最大列数[7] | 1012 |
| ORDER BY 子句中的列的最大总长度（以字节计）[7] | 32 677 |
| 最大子查询嵌套级别 | 存储器 |
| 单个语句中的最大子查询数 | 存储器 |
| 插入操作中的值的最大数目[7] | 1012 |
| 单个更新操作中的最大 SET 子句数[7] | 1012 |
| **表和视图** | |
| 表中的最大列数[7] | 1012 |
| 视图中的最大列数[1] | 5000 |
| 昵称引用的数据源表或视图中的最大列数 | 5000 |
| 分布键中的最大列数[5] | 500 |
| 行的最大长度（包括所有开销）[2 7] | 32 677 |
| 每个数据库分区的非分区表中的最大行数 | $128 \times 10^{10}$ |
| 每个数据库分区的数据分区中的最大行数 | $128 \times 10^{10}$ |
| 常规表空间中的每个数据库分区的最大表大小（以千兆字节计）[3 7] | 512 |
| 大型 DMS 表空间中的每个数据库分区的最大表大小（以千兆字节计）[7] | 16 384 |
| 单个表的最大数据分区数 | 32 767 |
| 最大表分区列数 | 16 |
| **表空间** | |
| LOB 对象的最大大小（以太字节计） | 4 |
| LF 对象的最大大小（以太字节计） | 2 |
| 数据库中的最大表空间数 | 32 768 |
| SMS 表空间中的最大表数 | 65 532 |
| 常规 DMS 表空间的最大大小（以千兆字节计）[3 7] | 512 |
| 大型 DMS 表空间的最大大小（以太字节计）[3 7] | 16 |
| 临时 DMS 表空间的最大大小（以太字节计）[37] | 16 |
| DMS 表空间中的最大表对象数[6] | 请参阅第 330 页的表 61 |
| 自动存储器数据库中的存储路径的最大数目 | 128 |
| 与自动存储器数据库关联的存储路径的最大长度（以字节计） | 175 |
| **触发器** | |
| 级联触发器的最大运行时深度 | 16 |
| **用户定义的类型** | |

*表 60. 数据库管理器限制 (续)*

| 描述 | 限制 |
|---|---|
| 结构化类型中的最大属性数 | 4082 |
| **注:**<br>1. 通过在 CREATE VIEW 语句中使用连接可获得此最大值。从这种视图中进行选择应遵循选择列表中大多数元素的限制。<br>2. BLOB、CLOB、LONG VARCHAR、DBCLOB 和 LONG VARGRAPHIC 列的实际数据未包括在此计数中。但是，关于该数据的位置的信息要占用行中的一些空间。<br>3. 显示的数字是体系结构限制和近似值。实际限制可能会小一些。<br>4. 实际值由 **max_connections** 和 **max_coordagents** 数据库管理器配置参数控制。<br>5. 这是体系结构限制。应将对索引键中大多数列的限制用作实际限制。<br>6. 请参阅表 61中的脚注 1。<br>7. 对于特定于页大小的值，请参阅表 61。<br>8. 它仅受包括所有开销在内的最长索引键（以字节计）限制。随着索引键部件的数目增多，每个键部件的最大长度将减小。<br>9. 最大值可能小一些，这取决于索引选项。 | |

*表 61. 数据库管理器特定于页大小的限制*

| 描述 | 4K 页大小限制 | 8K 页大小限制 | 16K 页大小限制 | 32K 页大小限制 |
|---|---|---|---|---|
| DMS 表空间中的最大表对象数[1] | 51 971[2]<br>53 212[3] | 53 299 | 53 747 | 54 264 |
| 表中的最大列数 | 500 | 1012 | 1012 | 1012 |
| 行的最大长度（包括所有开销） | 4005 | 8101 | 16 293 | 32 677 |
| 常规表空间中的每个数据库分区的最大表大小（以千兆字节计） | 64 | 128 | 256 | 512 |
| 大型 DMS 表空间中的每个数据库分区的最大表大小（以千兆字节计） | 2048 | 4096 | 8192 | 16 384 |
| 包括所有开销在内的最长索引键（以字节计） | 1024 | 2048 | 4096 | 8192 |
| SMS 表空间中的每个数据库分区的最大索引大小（以千兆字节计） | 2048 | 4096 | 8192 | 16 384 |
| 常规 DMS 表空间中的每个数据库分区的最大索引大小（以千兆字节计） | 64 | 128 | 256 | 512 |
| 大型 DMS 表空间中的每个数据库分区的最大索引大小（以千兆字节计） | 2048 | 4096 | 8192 | 16 384 |
| 常规 DMS 表空间的最大大小（以千兆字节计） | 64 | 128 | 256 | 512 |

*表 61. 数据库管理器特定于页大小的限制 (续)*

| 描述 | 4K 页大小限制 | 8K 页大小限制 | 16K 页大小限制 | 32K 页大小限制 |
|---|---|---|---|---|
| 大型 DMS 表空间的最大大小（以千兆字节计） | 2048 | 4096 | 8192 | 16 384 |
| 临时 DMS 表空间的最大大小（以太字节计） | 2 | 4 | 8 | 16 |
| 选择列表中的最大元素数 | 500 | 1012 | 1012 | 1012 |
| GROUP BY 子句中的最大列数 | 500 | 1012 | 1012 | 1012 |
| GROUP BY 子句中的列的最大总长度（以字节计） | 4005 | 8101 | 16 293 | 32 677 |
| ORDER BY 子句中的最大列数 | 500 | 1012 | 1012 | 1012 |
| ORDER BY 子句中的列的最大总长度（以字节计） | 4005 | 8101 | 16 293 | 32 677 |
| 插入操作中的值的最大数目 | 500 | 1012 | 1012 | 1012 |
| 单个更新操作中的最大 SET 子句数 | 500 | 1012 | 1012 | 1012 |
| 常规表空间的每页最大记录数 | 251 | 253 | 254 | 253 |
| 大型表空间的每页最大记录数 | 287 | 580 | 1165 | 2335 |

**注:**

1. 表对象包括表数据、索引、LONG VARCHAR 列、LONG VARGRAPHIC 列和 LOB 列。与表数据位于同一表空间中的表对象不会增大计数，不能使值更接近限制。但是，对于表对象所在的表空间中每个表的每种表对象类型，与表数据位于不同表空间中的每个表对象都使计数值增大 1，从而更接近限制。
2. 当扩展数据块大小为 2 页时。
3. 当扩展数据块大小不为 2 页时。

# 第 19 章 注册表变量和环境变量

## 环境变量和概要文件注册表

环境变量和注册表变量将控制 DB2 数据库环境。

可以使用“配置助手”（db2ca）来配置注册表变量和配置参数。

在引入 DB2 数据库概要文件注册表之前，在 Windows 工作站上更改环境变量（举例说明）要求您更改环境变量并重新启动。现在，除少数例外情况，您的环境均由 DB2 概要文件注册表中存储的注册表变量控制。在 UNIX 操作系统上，对于给定的实例来说，具有系统管理（SYSADM）权限的用户可更新该实例的注册表值。在 Windows 上，根据下列条件，更新概要文件注册表变量需要本地管理员权限或 SYSADM 权限:

- 如果启用了扩展安全性，那么 SYSADM 用户必须属于 DB2ADMNS 组。
- 如果未启用扩展安全性，那么只要在 Windows 注册表中为 SYSADM 用户授予了适当的许可权，这些用户就可以进行更新。

使用 db2set 命令来更新注册表变量，而不需重新启动；此信息会立即存储在概要文件注册表中。但是，所作的更改不会影响当前正在运行的 DB2 应用程序或用户。DB2 注册表将已更新的信息应用于更改后启动的 DB2 服务器实例和 DB2 应用程序。

**注**: 存在 DB2 环境变量 **DB2INSTANCE** 和 **DB2NODE**，它们可能未存储在 DB2 概要文件注册表中。在某些操作系统上，必须使用 set 命令才能更新这些环境变量。这些更改在下次重新启动系统之前一直有效。在 Linux 和 UNIX 平台上，可以使用 export 命令来代替 set 命令。

使用概要文件注册表允许集中控制环境变量。现在通过不同的概要文件提供了不同级别的支持。当使用 DB2 管理服务器时，还提供了对环境变量的远程管理。

有四个概要文件注册表:

- DB2 实例级概要文件注册表。大多数 DB2 环境变量都位于此注册表中。特定实例的环境变量设置保存在此注册表中。在此级别定义的值将覆盖在全局级的对应设置。
- DB2 全局级概要文件注册表。如果未对特定的实例设置环境变量，那么使用此注册表。对于属于 DB2 Enterprise Server Edition 的特定副本的所有实例来说，此注册表可视；安装路径中存在一个全局级概要文件。
- DB2 实例节点级概要文件注册表。在分区数据库环境中，此注册表级别包含特定于数据库分区的变量设置。在此级别定义的值将覆盖实例级和全局级的对应设置。
- DB2 实例概要文件注册表。此注册表包含与当前副本相关联的所有实例名的列表。每个安装都有它自己的列表。可通过运行 db2ilist 来查看系统上提供的所有实例的完整列表。

DB2 数据库系统通过按下列顺序检查注册表值和环境变量并解析它们来配置操作环境:

1. 使用 set 命令设置的环境变量。（或 UNIX 平台上的 export 命令。）
2. 使用实例节点级概要文件设置的注册表值（使用 `db2set -i instance_name nodenum` 命令）。

© Copyright IBM Corp. 1993, 2010

3. 使用实例级概要文件设置的注册表值（使用 db2set -i 命令）。
4. 使用全局级概要文件设置的注册表值（使用 db2set -g 命令）。

### 实例级概要文件注册表

使用分区数据库环境时，UNIX 和 Windows 会有一些差异。这些差异如以下示例所示。

假设分区数据库环境有三个以“红色”、“白色”和“蓝色”标识的物理数据库分区。在 UNIX 平台上，如果实例所有者从任一数据库分区运行下述命令:

```
db2set -i FOO=BAR
```

或者

```
db2set FOO=BAR          ("-i"是隐含的选项)
```

对于当前实例的所有节点（即“红色”、“白色”和“蓝色”节点），FOO 的值将是可视的。

在 UNIX 平台上，实例级概要文件注册表存储在 sqllib 目录下的文本文件中。在分区数据库环境下，sqllib 目录位于所有物理数据库分区共享的文件系统中。

在 Windows 平台上，如果用户从“红色”节点执行同一个命令，那么 FOO 的值只在当前实例的“红色”节点上可视。DB2 数据库管理器将实例级概要文件注册表存储在 Windows 注册表中。物理数据库分区之间没有共享。要设置所有物理计算机上的注册表变量，请使用 rah 命令，如下所示:

```
rah db2set -i FOO=BAR
```

rah 将在“红色”、“白色”和“蓝色”节点远程运行 db2set 命令。

可以使用 **DB2REMOTEPREG** 将没有实例的计算机上的注册表变量配置为引用具有实例的计算机上的注册表变量。这样可以有效地创建以下环境: 实例中的所有计算机共享具有实例的计算机上的注册表变量。

运用以上示例并假设“红色”节点为具有实例的计算机，即可通过下述操作将“白色”和“蓝色”计算机上的 **DB2REMOTEPREG** 设置为共享“红色”计算机上的注册表变量:

```
  (在红色机器上)不执行任何操作
(在白色和蓝色机器上)db2set DB2REMOTEPREG=\\red
```

在设置 **DB2REMOTEPREG** 后，不能更改其设置。

以下是 **DB2REMOTEPREG** 的工作方式:

当 DB2 数据库管理器在 Windows 上读取注册表变量时，它首先读取 **DB2REMOTEPREG** 值。如果设置了 **DB2REMOTEPREG**，那么将打开与 **DB2REMOTEPREG** 变量中指定的计算机名称对应的远程计算机上的注册表。将对指定的远程计算机重定向注册表变量的后续读取和更新。

访问远程注册表需要在目标计算机上运行“远程注册服务”。而且，用户登录帐户和所有 DB2 服务登录帐户对远程注册表有足够的访问权限。因此，要使用 **DB2REMOTEPREG**，应在 Windows 域环境下操作，这样才能授予域帐户必需的注册表访问权。

有一些 Microsoft Cluster Server（MSCS）注意事项。不应在 MSCS 环境中使用 **DB2REMOTEPREG**。当属于同一 MSCS 集群的所有计算机运行在 MSCS 配置（在此

配置下）中时，在集群注册表内维护注册表变量。因此，已在同一 MSCS 集群内的所有计算机之间共享它们，在此情况下没有必要使用 **DB2REMOTEPREG**。

当在数据库分区跨多个 MSCS 集群的多分区故障转移环境中运行时，因为具有实例的计算机的注册表变量位于集群注册表中，所以不能用 **DB2REMOTEPREG** 来指向具有实例的计算机。

## 声明、显示、更改、重新设置和删除注册表变量和环境变量

强烈建议所有特定的注册表变量都在 DB2 数据库概要文件注册表中定义。如果 DB2 变量是在注册表外部设置的，那么不能远程管理这些变量，并且必须在重新启动工作站后，这些变量值才会生效。

db2set 命令支持本地声明注册表变量和环境变量。

要显示该命令的帮助信息，使用:

```
db2set -?
```

要列示所有受支持的注册表变量的完整集合，使用:

```
db2set -lr
```

要列示当前实例或缺省实例的所有已定义的注册表变量，使用:

```
db2set
```

要列示概要文件注册表中所有定义的注册表变量，使用:

```
db2set -all
```

要显示一个注册表变量在当前实例或缺省实例中的值，使用:

```
db2set registry_variable_name
```

要显示一个注册表变量在所有级别的值，使用:

```
db2set registry_variable_name -all
```

要在当前实例或缺省实例中更改一个注册表变量，使用:

```
db2set registry_variable_name=new_value
```

要更改该实例中所有数据库的注册表变量缺省值，使用:

```
db2set registry_variable_name=new_value
       -i instance_name
```

要更改实例中特定数据库分区的注册表变量缺省值，使用:

```
db2set registry_variable_name=new_value
 -i instance_name database_partition_number
```

要更改系统中属于特定安装的所有实例的注册表变量缺省值，使用:

```
db2set registry_variable_name=new_value -g
```

如果使用聚集注册表变量（例如 **DB2_WORKLOAD**）来配置 SAP 环境的注册表变量，那么可以使用下列命令来设置该变量:

```
db2set DB2_WORKLOAD=SAP
```

如果使用“轻量级目录访问协议”（LDAP），那么可以使用下列命令在 LDAP 中设置注册表变量:

- 要在 LDAP 中设置用户级的注册表变量，使用:

  ```
  db2set -ul
  ```

- 要在 LDAP 中设置全局级的注册表变量，使用:

  ```
  db2set -gl user_name
  ```

当在 LDAP 环境中运行时，可以对 DB2 注册表变量值作如下设置: 作用域对属于某个目录分区或 Windows 域的所有服务器和所有用户是全局的。当前，在 LDAP 全局级只能设置两个 DB2 注册表变量: **DB2LDAP_KEEP_CONNECTION** 和 **DB2LDAP_SEARCH_SCOPE**。

例如，要在 LDAP 中的全局级设置搜索范围值，使用:

```
db2set -gl db2ldap_search_scope = value
```

其中 value 可以是“local”（局部）、“domain”（域）或“global”（全局）。

**注:**

1. 当两个或多个用户使用 db2set 命令同时（或接近于同时）更新 DB2 profile.env 文件时，profile.env 文件的大小将减少为零。而且 db2set -all 的输出显示不一致的值。
2. 用于设置 DB2 注册表变量（这些变量适用于与 DB2 ESE 的相同安装有关的所有实例）的 -g 选项与 LDAP 全局级专用的 -gl 选项之间存在差别。
3. 在 LDAP 环境中运行时，用户级注册表变量仅在 Windows 上受支持。
4. 用户级的变量设置包含用户特定变量设置。对用户级的任何更改都会写入 LDAP 目录。
5. 在同一命令中不能同时使用参数“-i”、“-g”、“-gl”和“-ul”。
6. 某些变量将始终缺省设置为全局级概要文件（全局意味着变量在同一个 DB2 副本上运行的所有实例间共享）。不能在实例级或数据库分区级概要文件（如 **DB2SYSTEM** 和 **DB2INSTDEF**）对它们进行设置。
7. 在 UNIX 上，必须具有系统管理（SYSADM）权限，才能更改实例的注册表值。只有具有 root 用户权限的用户才能更改全局级注册表中的参数。

要将实例的注册表变量复位回“全局概要文件注册表”中的缺省值，请使用:

```
db2set -r registry_variable_name
```

要将实例中数据库分区的注册表变量复位回“全局概要文件注册表”中的缺省值，请使用:

```
db2set -r registry_variable_name database_partition_number
```

要删除一个变量在特定级别的值，可使用相同的命令语法来设置该变量但不对该变量值指定任何内容。例如，要删除该变量在数据库分区级的设置，请输入:

```
db2set registry_variable_name= -i instance_name
   database_partition_number
```

要删除变量的值并限制变量的用途，如果它是在更高的概要文件级定义的，那么输入:

```
db2set registry_variable_name= -null -r instance_name
```

此命令将删除指定参数的设置，并限制高级别的概要文件（在本示例中为 DB2 全局级概要文件）更改此变量的值。但指定的变量仍可由低级别的概要文件（在本示例中为 DB2 数据库分区级概要文件）设置。

## 在 Windows 上设置环境变量

Windows 操作系统有一个系统环境变量 **DB2INSTANCE**，此变量只能在概要文件注册表外进行设置；但不要求您设置 **DB2INSTANCE**。可以在全局级概要文件中设置 DB2 概要文件注册表变量 **DB2INSTDEF**，以指定在未定义 **DB2INSTANCE** 的情况下要使用的实例名。

Windows 上的 DB2 Enterprise Server Edition服务器有两个系统环境变量 **DB2INSTANCE** 和 **DB2NODE**，这两个变量只能在概要文件注册表外进行设置。不要求您设置 **DB2INSTANCE**。可以在全局级概要文件中设置 DB2 概要文件注册表变量 **DB2INSTDEF**，以指定在未定义 **DB2INSTANCE** 的情况下要使用的实例名。

**DB2NODE** 环境变量用于将请求路由至计算机内的目标逻辑节点。必须在发出该应用程序或命令的会话中而不是在 DB2 概要文件注册表中设置此环境变量。如果未设置此变量，那么目标逻辑节点缺省为在该计算机上定义为零（0）的逻辑节点。

要确定环境变量的设置，可使用 echo 命令。例如，要检查 **DB2PATH** 环境变量的值，请输入:

```
echo %db2path%
```

可按如下所示设置 DB2 环境变量 **DB2INSTANCE** 和 **DB2NODE**（在此描述中使用 **DB2INSTANCE**）:

- 右键单击“我的电脑”并选择“属性”。
- 选择“高级”选项卡，单击“环境变量”并执行下列操作:
  1. 如果 **DB2INSTANCE** 变量不存在:
     a. 单击“新建”。
     b. 在“变量名”字段中填写 **DB2INSTANCE**。
     c. 在“变量值”字段填写实例名，例如，db2inst。
  2. 如果 **DB2INSTANCE** 变量已经存在，那么追加新的值:
     a. 选择 **DB2INSTANCE** 环境变量。
     b. 将“值”字段更改为实例名，例如 db2inst。
  3. 重新启动系统，以使这些更改生效。

**注**: 也可以在会话（进程）级别设置环境变量 **DB2INSTANCE**。例如，如果要启动另一个 DB2 实例 TEST，在命令窗口中发出下列命令:

```
set DB2INSTANCE=TEST
   db2start
```

当在 C shell 中工作时，在命令窗口中发出下列命令:

```
setenv DB2INSTANCE TEST
```

概要文件注册表位置如下:

- “DB2 实例级概要文件注册表”位于 Windows 操作系统注册表中，其路径为:

\HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\PROFILES*instance_name*

**注:** *instance_name* 是 DB2 实例的名称。

- “DB2 全局级概要文件注册表”位于 Windows 注册表中，其路径为:

  \HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\GLOBAL_PROFILE

- “DB2 实例节点级概要文件注册表”位于 Windows 注册表中，其路径为:

  ...\SOFTWARE\IBM\DB2\PROFILES*instance_name*\NODES*node_number*

  **注:** *instance_name* 和 *node_number* 特定于您正在使用的数据库分区。

- 不需要“DB2 实例概要文件注册表”。对于系统中的每个 DB2 实例，在下列路径中创建一个键:

  \HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\PROFILES*instance_name*

  实例列表可通过对 PROFILES 键中的键进行计数获取。

## 在 Linux 和 UNIX 操作系统上设置环境变量

在 UNIX 操作系统上，必须设置系统环境变量 **DB2INSTANCE**。

提供脚本 db2profile（对于 Bourne 或 Korn shell 程序）和 db2cshrc（对于 C shell）作为示例，以帮助您设置数据库环境。可以在 insthome/sqllib 中找到这些文件，其中 insthome 是实例所有者的主目录。

这些脚本包括对下列各项的说明:

- 使用下列目录来更新用户的路径:
  - insthome/sqllib/bin
  - insthome/sqllib/adm
  - insthome/sqllib/misc
- 将 **DB2INSTANCE** 设置为用于执行的缺省本地 instance_name。

**注:** 除 PATH 和 **DB2INSTANCE** 外，其他所有受支持的变量都必须在 DB2 概要文件注册表中设置。要设置 DB2 数据库管理器不支持的变量，请在脚本文件 userprofile 和 usercshrc 中对它们进行定义。

实例所有者或 SYSADM 用户可以为一个实例的所有用户定制这些脚本。或者，用户可以复制和定制脚本，然后直接调用脚本或将它添加至它们的 .profile 或 .login 文件。

要更改当前会话的环境变量，发出类似于以下的命令:

- 对于 Korn shell 程序:

```
DB2INSTANCE=<inst1>
   export DB2INSTANCE
```

- 对于 Bourne shell:

```
export DB2INSTANCE=<inst1>
```

- 对于 C shell:

```
setenv DB2INSTANCE <inst1>
```

为了正确管理 DB2 概要文件注册表，在 UNIX 操作系统上必须遵循下列文件所有权规则。

- “DB2 实例级概要文件注册表”文件位于:

  INSTHOME/sqllib/profile.env

  此文件的访问许可权和所有权应该是:

  ```
  -rw-rw-r-- <db2inst1> <db2iadm1> profile.env
  ```

  其中 <db2inst1> 是实例所有者，而 <db2iadm1> 是实例所有者的组。

  INSTHOME 是实例所有者的主路径。
- “DB2 全局级概要文件注册表”位于:

  <installation path>/default.env

  适用于所有 Linux 和 UNIX 平台。

  此文件的访问许可权和所有权应该是:

  -rw-rw-r-- root <group> default.env

  其中 <group> 是组标识 0 的组名；例如，在 AIX 上，它为“system”。

  要修改 root 用户安装中的全局注册表变量，您必须作为具有 root 用户权限的用户登录。
- “DB2 实例节点级概要文件注册表”位于:

  ```
  INSTHOME/sqllib/nodes/<node_number>.env
  ```

  该目录和此文件的访问许可权和所有权应该是:

  ```
  drwxrwsr-w <Instance_Owner> <Instance_Owner_Group> nodes
  ```

  ```
  -rw-rw-r-- <Instance_Owner> <Instance_Owner_Group> <node_number>.env
  ```

  INSTHOME 是实例所有者的主路径。
- “DB2 实例概要文件注册表”位于:

  <installation path>/profiles.reg

  适用于所有 Linux 和 UNIX 平台。

  此文件的访问许可权和所有权应该是:

  ```
  -rw-r--r-- root system profiles.reg
  ```

## 设置当前实例环境变量

当运行命令以启动或停止实例的数据库管理器时，DB2 将该命令应用于当前实例。DB2 按如下所示确定当前实例:

- 如果为当前会话设置了 **DB2INSTANCE** 环境变量，那么其值为当前实例。要设置 **DB2INSTANCE**，请输入:

  ```
  set db2instance=<new_instance_name>
  ```
- 如果没有为当前会话设置 **DB2INSTANCE**，那么 DB2 数据库管理器使用系统环境变量中 **DB2INSTANCE** 环境变量的值。在 Windows 上，在“系统环境”注册表中设置系统环境变量。

- 如果根本没有设置 **DB2INSTANCE**，那么 DB2 数据库管理器使用注册表变量 **DB2INSTDEF**。

  要在注册表的全局级设置 **DB2INSTDEF** 注册表变量，请输入:

  ```
  db2set db2instdef=<new_instance_name> -g
  ```

要确定哪一个实例适用于当前会话，请输入:

```
db2 get instance
```

## 聚集注册表变量

聚集注册表变量允许几个注册表变量组成一个配置，该配置可由另一个注册表变量名标识。作为该组一部分的每个注册表变量都具有预定义的设置。为聚集注册表变量提供的值被解释为声明几个注册表变量。

聚集注册表变量的作用是使大量操作目标的注册表配置变得容易。

唯一有效的聚集注册表变量是 **DB2_WORKLOAD**。

此变量的有效值为:

- 1C
- CM
- COGNOS_CS
- FILENET_CM
- INFOR_ERP_LN
- MAXIMO
- MDM
- SAP
- TPM
- WAS
- WC
- WP

通过聚集注册表变量隐式配置的任何注册表变量也可以显式定义。显式设置先前通过使用聚集注册表变量给定了值的注册表变量在进行性能或诊断测试时非常有用。显式设置通过聚集隐式配置的变量称为“覆盖”变量。

如果尝试使用聚集注册表变量来修改显式设置的注册表变量，那么将会发出警告并保留显式设置的值。此警告告诉您将保留显式的值。如果首先使用聚集注册表变量，然后指定显式注册表变量，那么不会发出警告。

除非您显式地请求每个变量，否则不会显示通过设置聚集注册表变量来配置的任何一个注册表变量。当查询聚集注册表变量时，只会显示指定给该变量的值。大部分用户应该不会关心每个单独变量的值。

以下示例显示使用聚集注册表变量与显式设置注册表变量之间的交互。例如，您可能已将 **DB2_WORKLOAD** 聚集注册表变量设置为 SAP 并已将 **DB2_SKIPDELETED** 注册表变量覆盖为 NO。通过输入 db2set，您将接收到下列结果:

```
DB2_WORKLOAD=SAP
DB2_SKIPDELETED=NO
```

在另一种情况下，您可能已设置 **DB2ENVLIST**、已将 **DB2_WORKLOAD** 聚集注册表变量设置为 SAP 并已将 **DB2_SKIPDELETED** 注册表变量覆盖为 NO。（假定 **DB2_SKIPDELETED** 注册表变量是构成 SAP 环境的组的一部分。）另外，那些通过设置聚集注册表变量而自动配置的注册表变量将在其值旁边的方括号中显示聚集的名称。**DB2_SKIPDELETED** 注册表变量将显示 NO 值并将在它的值旁边显示 [O]。

当您不再需要与 **DB2_WORKLOAD** 相关联的配置时，可以通过删除聚集注册表变量的值来禁用组中每个注册表变量的隐式值，命令如下:

```
db2set DB2_WORKLOAD=
```

在删除 **DB2_WORKLOAD** 聚集注册表变量值后，请重新启动数据库。在重新启动数据库后，由该聚集注册表变量隐式配置的注册表变量不再有效。用来删除聚集注册表变量的值的方法与删除单个注册表变量的方法相同。

删除聚集注册表变量的值不会删除已显式设置的注册表变量的值。即使该注册表变量是被禁用的组定义的成员也没关系。注册表变量的显式设置会保留下来。

您可能需要查看作为 **DB2_WORKLOAD** 聚集注册表变量的成员的每个注册表变量的值。例如，您可能想查看在将 **DB2_WORKLOAD** 配置为 SAP 的情况下将使用的值。要查找 **DB2_WORKLOAD**=SAP 时将使用的值，请运行 `db2set -gd DB2_WORKLOAD=SAP`。

## DB2 注册表变量和环境变量

DB2 数据库产品提供了许多注册表变量和环境变量，这可能是在启动和运行时所要了解的内容。

要查看所有受支持的注册表变量的列表，请执行以下命令:

```
db2set -lr
```

要更改当前实例或缺省实例中变量的值，请执行以下命令:

```
db2set registry_variable_name=new_value
```

DB2 环境变量 **DB2INSTANCE**、**DB2NODE**、**DB2PATH** 和 **DB2INSTPROF** 是否存储在 DB2 概要文件注册表中取决于操作系统。要更新这些环境变量，可使用 set 命令。这些更改仅在本地（当前）命令提示符中有效，并且在系统下次重新引导之前将一直有效。在 Linux 和 UNIX 操作系统上，可以使用 export 命令来代替 set 命令。

在执行 db2start 命令之前，必须设置已更改注册表变量的值。

**注:** 如果注册表变量要求布尔值自变量，那么值 YES、1、TRUE 和 ON 是等同的，并且值 NO、0、FALSE 和 OFF 也是等同的。对于任何变量，可以指定任何适当的等价值。

下表列示了每种类别的所有注册表变量:

*表 62. 注册表变量和环境变量摘要*

| 变量类别 | 注册表或环境变量名 |
| --- | --- |
| 常规 | **DB2ACCOUNT**<br>**DB2BIDI**<br>**DB2_CAPTURE_LOCKTIMEOUT**<br>**DB2CODEPAGE**<br>**DB2_COLLECT_TS_REC_INFO**<br>**DB2_CONNRETRIES_INTERVAL**<br>**DB2CONSOLECP**<br>**DB2COUNTRY**<br>**DB2DBDFT**<br>**DB2DBMSADDR**<br>**DB2DISCOVERYTIME**<br>**DB2FFDC**<br>**DB2FODC**<br>**DB2_FORCE_APP_ON_MAX_LOG**<br>**DB2GRAPHICUNICODESERVER**<br>**DB2INCLUDE**<br>**DB2INSTDEF**<br>**DB2INSTOWNER**<br>**DB2_LIC_STAT_SIZE**<br>**DB2LOCALE**<br>**DB2_MAX_CLIENT_CONNRETRIES**<br>**DB2_OBJECT_TABLE_ENTRIES**<br>**DB2_SYSTEM_MONITOR_SETTINGS**<br>**DB2TERRITORY**<br>**DB2_VIEW_REOPT_VALUES** |
| 系统环境 | **DB2_ALTERNATE_GROUP_LOOKUP**<br>**DB2CONNECT_ENABLE_EURO_CODEPAGE**<br>**DB2CONNECT_IN_APP_PROCESS**<br>**DB2_COPY_NAME**<br>**DB2DBMSADDR**<br>**DB2_DIAGPATH**<br>**DB2DOMAINLIST**<br>**DB2ENVLIST**<br>**DB2INSTANCE**<br>**DB2INSTPROF**<br>**DB2LDAPSecurityConfig**<br>**DB2LIBPATH**<br>**DB2LOGINRESTRICTIONS**<br>**DB2NODE**<br>**DB2OPTIONS**<br>**DB2_PARALLEL_IO**<br>**DB2PATH**<br>**DB2PROCESSORS**<br>**DB2RCMD_LEGACY_MODE**<br>**DB2SYSTEM**<br>**DB2_UPDDBCFG_SINGLE_DBPARTITION**<br>**DB2_USE_PAGE_CONTAINER_TAG**<br>**DB2_WORKLOAD** |

*表 62. 注册表变量和环境变量摘要 (续)*

| 变量类别 | 注册表或环境变量名 |
| --- | --- |
| 通信 | **DB2CHECKCLIENTINTERVAL**<br>**DB2COMM**<br>**DB2FCMCOMM**<br>**DB2_FORCE_NLS_CACHE**<br>**DB2RSHCMD**<br>**DB2RSHTIMEOUT**<br>**DB2SORCVBUF**<br>**DB2SOSNDBUF**<br>**DB2TCP_CLIENT_CONTIMEOUT**<br>**DB2TCP_CLIENT_RCVTIMEOUT**<br>**DB2TCPCONNMGRS** |
| 命令行 | **DB2BQTIME**<br>**DB2BQTRY**<br>**DB2_CLP_EDITOR**<br>**DB2_CLP_HISTSIZE**<br>**DB2_CLPPROMPT**<br>**DB2IQTIME**<br>**DB2RQTIME** |
| 分区数据库环境 | **DB2CHGPWD_EEE**<br>**DB2_FCM_SETTINGS**<br>**DB2_NUM_FAILOVER_NODES**<br>**DB2_PARTITIONEDLOAD_DEFAULT**<br>**DB2PORTRANGE** |
| 查询编译器 | **DB2_ANTIJOIN**<br>**DB2_INLIST_TO_NLJN**<br>**DB2_LIKE_VARCHAR**<br>**DB2_MINIMIZE_LISTPREFETCH**<br>**DB2_NEW_CORR_SQ_FF**<br>**DB2_OPT_MAX_TEMP_SIZE**<br>**DB2_REDUCED_OPTIMIZATION**<br>**DB2_SELECTIVITY**<br>**DB2_SQLROUTINE_PREPOPTS** |

*表 62. 注册表变量和环境变量摘要 (续)*

| 变量类别 | 注册表或环境变量名 |
| --- | --- |
| 性能 | **DB2_ALLOCATION_SIZE**<br>**DB2_APM_PERFORMANCE**<br>**DB2ASSUMEUPDATE**<br>**DB2_ASYNC_IO_MAXFILOP**<br>**DB2_AVOID_PREFETCH**<br>**DB2BPVARS**<br>**DB2CHKPTR**<br>**DB2CHKSQLDA**<br>**DB2_EVALUNCOMMITTED**<br>**DB2_EXTENDED_IO_FEATURES**<br>**DB2_EXTENDED_OPTIMIZATION**<br>**DB2_HASH_JOIN**<br>**DB2_IO_PRIORITY_SETTING**<br>**DB2_KEEP_AS_AND_DMS_CONTAINERS_OPEN**<br>**DB2_KEEPTABLELOCK**<br>**DB2_LARGE_PAGE_MEM**<br>**DB2_LOGGER_NON_BUFFERED_IO**<br>**DB2MAXFSCRSEARCH**<br>**DB2_MAX_INACT_STMTS**<br>**DB2_MAX_NON_TABLE_LOCKS**<br>**DB2_MDC_ROLLOUT**<br>**DB2MEMDISCLAIM**<br>**DB2MEMMAXFREE**<br>**DB2_MEM_TUNING_RANGE**<br>**DB2_MMAP_READ**<br>**DB2_MMAP_WRITE**<br>**DB2_NO_FORK_CHECK**<br>**DB2NTMEMSIZE**<br>**DB2NTNOCACHE**<br>**DB2NTPRICLASS**<br>**DB2NTWORKSET**<br>**DB2_OVERRIDE_BPF**<br>**DB2_PINNED_BP**<br>**DB2PRIORITIES**<br>**DB2_RESOURCE_POLICY**<br>**DB2_RCT_FEATURES**<br>**DB2_SET_MAX_CONTAINER_SIZE**<br>**DB2_SKIPDELETED**<br>**DB2_SKIPINSERTED**<br>**DB2_SMS_TRUNC_TMPTABLE_THRESH**<br>**DB2_SORT_AFTER_TQ**<br>**DB2_SELUDI_COMM_BUFFER**<br>**DB2_TRUSTED_BINDIN**<br>**DB2_USE_ALTERNATE_PAGE_CLEANING**<br>**DB2_USE_FAST_PREALLOCATION**<br>**DB2_USE_IOCP** |

*表 62. 注册表变量和环境变量摘要 (续)*

| 变量类别 | 注册表或环境变量名 |
| --- | --- |
| 其他 | **DB2ADMINSERVER**<br>**DB2_ATS_ENABLE**<br>**DB2AUTH**<br>**DB2CLIINIPATH**<br>**DB2_COMMIT_ON_EXIT**<br>**DB2_CREATE_DB_ON_PATH**<br>**DB2DEFPREP**<br>**DB2_DISABLE_FLUSH_LOG**<br>**DB2_DISPATCHER_PEEKTIMEOUT**<br>**DB2_DJ_INI**<br>**DB2DMNBCKCTLR**<br>**DB2_DOCHOST**<br>**DB2_DOCPORT**<br>**DB2_ENABLE_AUTOCONFIG_DEFAULT**<br>**DB2_ENABLE_LDAP**<br>**DB2_EVMON_EVENT_LIST_SIZE**<br>**DB2_EVMON_STMT_FILTER**<br>**DB2_EXTSECURITY**<br>**DB2_FALLBACK**<br>**DB2_FMP_COMM_HEAPSZ**<br>**DB2_GRP_LOOKUP**<br>**DB2_HADR_BUF_SIZE**<br>**DB2_HADR_NO_IP_CHECK**<br>**DB2_HADR_PEER_WAIT_LIMIT**<br>**DB2_HADR_SORCVBUF**<br>**DB2_HADR_SOSNDBUF**<br>**DB2LDAP_BASEDN**<br>**DB2LDAPCACHE**<br>**DB2LDAP_CLIENT_PROVIDER**<br>**DB2LDAPHOST**<br>**DB2LDAP_KEEP_CONNECTION**<br>**DB2LDAP_SEARCH_SCOPE**<br>**DB2_LOAD_COPY_NO_OVERRIDE**<br>**DB2LOADREC**<br>**DB2LOCK_TO_RB**<br>**DB2_MAP_XML_AS_CLOB_FOR_DLC**<br>**DB2_MAX_LOB_BLOCK_SIZE**<br>**DB2_MEMORY_PROTECT**<br>**DB2NOEXITLIST**<br>**DB2_NUM_CKPW_DAEMONS**<br>**DB2_OPTSTATS_LOG**<br>**DB2REMOTEPREG**<br>**DB2_RESOLVE_CALL_CONFLICT**<br>**DB2ROUTINE_DEBUG**<br>**DB2SATELLITEID**<br>**DB2_SERVER_CONTIMEOUT**<br>**DB2_SERVER_ENCALG**<br>**DB2SORT**<br>**DB2_THREAD_SUSPENSION** |

*表 62. 注册表变量和环境变量摘要 (续)*

| 变量类别 | 注册表或环境变量名 |
| --- | --- |
| 其他 | **DB2_TRUNCATE_REUSESTORAGE**<br>**DB2_USE_DB2JCCT2_JROUTINE**<br>**DB2_UTIL_MSGPATH**<br>**DB2_VENDOR_INI**<br>**DB2_XBSA_LIBRARY** |

## 常规注册表变量

### DB2ACCOUNT

- 操作系统: 所有操作系统
- 缺省值: NULL
- 此变量定义发送至远程主机的记帐字符串。有关详细信息，参阅 《DB2 Connect 用户指南》。

### DB2BIDI

- 操作系统: 所有操作系统
- 缺省值: NO; 值: YES 或 NO
- 此变量启用双向支持，并且 **DB2CODEPAGE** 变量用于声明要使用的代码页。

### DB2_CAPTURE_LOCKTIMEOUT

- 操作系统: 所有操作系统
- 缺省值: NULL; 值: ON 或 NULL
- 此变量指定在发生锁定超时的情况下日志记录关于锁定超时的描述性信息。日志记录的信息标识: 导致锁定超时的锁定争用中所涉及的关键应用程序、关于这些应用程序在锁定超时的情况下正在运行的对象的详细信息，以及关于导致争用的锁定的详细信息。将同时捕获关于锁定请求者（接收锁定超时错误的应用程序）和当前锁定所有者的信息。对于每个锁定超时，将编写一个文本报告并将其存储在文件中。

  这些文件是使用以下命名约定创建的: db2locktimeout.*par*.*AGENTID*.*yyyy-mm-dd-hh-mm-ss*，其中 *par* 是数据库分区号; *AGENTID* 是代理程序标识; *yyyy-mm-dd-hh-mm-ss* 是时间戳记，由年、月、日、小时、分钟和秒组成。在非分区数据库环境中，*par* 设置为 0。

  文件的位置取决于在 **diagpath** 数据库配置参数中设置的值。如果未设置 **diagpath**，那么文件位于下面其中一个目录中:

  – 在 Windows 环境中:
    - 如果未设置 **DB2INSTPROF** 环境变量，那么信息将写入 *x*:*SQLLIB**DB2INSTANCE*，其中 *x* 是驱动器引用，*SQLLIB* 是对 **DB2PATH** 注册表变量指定的目录，而 *DB2INSTANCE* 是实例所有者的名称。
    - 如果设置了 **DB2INSTPROF** 环境变量，那么信息将写入 *x*:*DB2INSTPROF**DB2INSTANCE*，其中 *x* 是驱动器引用，*DB2INSTPROF* 是实例概要文件目录的名称，而 *DB2INSTANCE* 是实例所有者的名称。

- 如果您将 **DB2INSTPROF** 环境变量设置为一个新位置，那么必须确保它包含适当的文件和文件夹以运行该实例。这可能会要求您将所有文件和文件夹从先前位置复制到新位置。

– 在 Linux 和 UNIX 环境中：信息将写入 *INSTHOME*/sqllib/db2dump，其中 *INSTHOME* 是实例的主目录。

当您不再需要锁定超时报告文件时，将这些文件删除。由于报告文件与其他诊断日志位于相同位置中，因此在允许该目录变满的情况下，DB2 系统可能会关闭。如果需要保留某些锁定超时报告文件，那么将它们移至 DB2 日志的存储目录或文件夹之外的目录或文件夹。

**DB2CODEPAGE**

- 操作系统：所有操作系统
- 缺省值：从语言标识中派生，由操作系统指定。
- 此变量指定为数据库客户机应用程序呈示给 DB2 的数据的代码页。除非在 DB2 文档中明确说明或 DB2 服务要求这样做，否则用户不应设置 **DB2CODEPAGE**。将 **DB2CODEPAGE** 设置为不受操作系统支持的值可能会产生意外结果。通常，因为 DB2 会自动从操作系统派生该代码页信息，所以不需要设置 **DB2CODEPAGE**。

  **注**：由于 Windows 不报告 Unicode 代码页（在 Windows 区域设置中），而不是报告 ANSII 代码页，所以 Windows 应用程序不能用作 Unicode 客户机。要覆盖此行为，将 **DB2CODEPAGE** 注册表变量设置为 1208（表示 Unicode 代码页）以将应用程序用作 Unicode 应用程序。

**DB2_COLLECT_TS_REC_INFO**

- 操作系统：所有操作系统
- 缺省值：ON；值：ON 或 OFF
- 此变量指定在前滚表空间时，DB2 是否将处理所有日志文件，无论日志文件是否包含会影响表空间的日志记录。要跳过已知不包含将影响表空间的任何日志记录的日志文件，应将此变量设置为 ON。在创建和使用日志文件之前，必须设置 **DB2_COLLECT_TS_REC_INFO** ，以便收集跳过日志文件的所需要的信息。

**DB2_CONNRETRIES_INTERVAL**

- 操作系统：所有操作系统
- 缺省值：未设置；值：整数秒数
- 此变量指定客户机自动重新路由功能的连续连接重试之间的休眠时间（以秒计）。可以将此变量与 **DB2_MAX_CLIENT_CONNRETRIES** 配合使用以配置客户机自动重新路由重试行为。

  如果设置了 **DB2_MAX_CLIENT_CONNRETRIES**，但未设置 **DB2_CONNRETRIES_INTERVAL**，那么 **DB2_CONNRETRIES_INTERVAL** 缺省为 30。如果未设置 **DB2_MAX_CLIENT_CONNRETRIES**，但设置了 **DB2_CONNRETRIES_INTERVAL**，那么 **DB2_MAX_CLIENT_CONNRETRIES** 缺省为 10。如果既未设置 **DB2_MAX_CLIENT_CONNRETRIES** 也未设置 **DB2_CONNRETRIES_INTERVAL**，那么客户机自动重新路由功能将恢复为采用缺省行为，即重复地重新尝试连接数据库，最多 10 分钟。

**DB2CONSOLECP**

- 操作系统: Windows
- 缺省值: NULL; 值: 所有有效的代码页值
- 指定用于显示 DB2 消息文本的代码页。当指定了此值时，它会覆盖操作系统代码页设置。

**DB2COUNTRY**

- 操作系统: Windows
- 缺省值: NULL; 值: 所有有效数字的国家或地区、地域或区域代码
- 此变量指定客户机应用程序的国家或地区、地域或区域代码。当指定了此值时，它会覆盖操作系统设置。

**注**: 不推荐使用 **DB2COUNTRY**，将来的发行版中可能会将其除去。请改为使用 DB2TERRITORY，它与 **DB2COUNTRY** 接受相同的值。

**DB2DBDFT**

- 操作系统: 所有操作系统
- 缺省值: NULL
- 此变量指定要用于隐式连接的数据库的数据库别名。如果应用程序没有数据库连接，但发出了 SQL 或 XQuery 语句，那么在缺省数据库中定义了 **DB2DBDFT** 环境变量的情况下，将建立隐式连接。

**DB2DBMSADDR**

- 操作系统: Windows 32 位
- 缺省值: 0x20000000; 值: 0x20000000 到 0xB0000000，增量为 0x10000
- 此变量指定十六进制格式的缺省数据库管理器共享内存地址。如果 db2start 因共享内存地址冲突而失败，那么可修改此注册表变量，以强制数据库管理器实例在另一个地址分配其共享内存。

**DB2DISCOVERYTIME**

- 操作系统: Windows
- 缺省值: 40 秒，最小值: 20 秒
- 此变量指定 SEARCH 发现将在 DB2 系统中搜索的时间。

**DB2_EXPRESSION_RULES**

- 操作系统: 所有操作系统
- 缺省值: 空; 值: RAISE_ERROR_PERMIT_SKIP 或 RAISE_ERROR_PERMIT_DROP
- **DB2_EXPRESSION_RULES** 注册表变量的设置控制 DB2 优化器如何确定涉及 RAISE_ERROR 函数的查询的存取方案。RAISE_ERROR 函数的缺省行为是不可将任何过滤推送到包含此函数的表达式范围之外。结果，在表访问期间不会应用任何谓词，从而导致表达式计算过量、锁定过量和查询性能不佳。

  在特定情况下，此行为限制过于严格。取决于应用程序的特定业务需求，如果在应用 RAISE_ERROR 之前应用谓词和连接，这可能也无关紧要。例如，在行级别的安全实现上下文中，表达式通常为以下格式:

```
CASE WHEN <conditions for validatin access to this row>
     THEN NULL
     ELSE RAISE_ERROR(...)
  END
```

应用程序可能仅与验证对查询所选行的访问相关，而与验证对表中每一行的访问无关。因此，可在基本表访问中应用谓词，且只需在进行了所有过滤后执行包含 RAISE_ERROR 的表达式。在此情况下，值 **DB2_EXPRESSION_RULES**=RAISE_ERROR_PERMIT_SKIP 可能会恰当一些。

另一种情况是在 COLUMN LEVEL 安全上下文中。在此情况下，表达式通常为以下格式:

```
CASE WHEN <conditions for validating access to this row and column>
     THEN <table.column>
     ELSE RAISE_ERROR(...)
  END
```

此时，应用程序可能只想在以下情况下让系统报告错误: 用户尝试接收特定行的数据，而列包含不允许用户检索的值。在此情况下，如果此特定列由谓词或列函数使用或者作为查询的输出返回，那么，设置 **DB2_EXPRESSION_RULES**=RAISE_ERROR_PERMIT_DROP 只会导致对包含 RAISE_ERROR 函数的表达式进行求值。

**DB2FFDC**

- 操作系统: 所有操作系统
- 缺省值: ON; 值: ON 或 CORE:OFF
- 此变量提供了取消激活核心文件生成的功能。缺省情况下，将此注册表变量设置为 ON。如果未设置此注册表变量，或者设置为除 CORE:OFF 以外的值，那么 DB2 数据库系统异常终止时可能会生成核心文件。

  核心文件是在 **diagpath** 目录中创建的用于问题确定的文件，它们包含终止 DB2 进程的整个进程映像。应注意可用的文件系统空间，因为核心文件可能非常庞大。该大小取决于 DB2 数据库系统配置和发生问题时进程的状态。

  在 Linux 操作系统上，缺省核心文件大小限制为 0（即，`ulimit -c`）。对于此设置，不会生成核心文件。在 Linux 平台上，要允许生成核心文件，应将该值设置为无限制。

**注**: 版本 9.5 中不推荐使用 **DB2FFDC**，在以后的发行版中会将其除去。新注册表变量 **DB2FODC** 包含了 **DB2FFDC** 的功能。

**DB2FODC**

- 操作系统: 所有操作系统
- 缺省值: 所有首次出现数据收集（FODC）参数的并置（请参阅下面的内容）
  - 对于 Linux 和 UNIX: "CORELIMIT=*val* DUMPCORE=ON DUMPDIR=*diagpath*"
  - 对于 Windows: "DUMPCORE=ON DUMPDIR=*diagpath*"

  请注意，参数之间用空格分隔。

- 此注册表变量控制在 FODC 中使用的一组与故障诊断相关的参数。使用 **DB2FODC** 控制中断情况下数据收集的不同方面。

  此注册表变量在 DB2 实例启动期间读取一次。要对 FODC 参数进行联机更新，请使用 db2pdcfg 工具。使用 **DB2FODC** 注册表变量使配置在重新引导期间保持有效。不需要指定所有参数，也不需要按特定顺序指定参数。将对未指定的任何参数指定缺省值。例如，如果不想转储核心文件，但您需要其他参数的缺省行为，那么可发出以下命令:

  ```
  db2set DB2FODC="DUMPCORE=OFF"
  ```

  参数:

  **CORELIMIT**

  - 操作系统: Linux 和 UNIX
  - 缺省值: 当前 ulimit 设置; 值: 0 到 unlimited
  - 此选项指定所创建核心文件的最大大小（按字节计）。此值将覆盖当前核心文件大小限制设置。应注意可用的文件系统空间，因为核心文件可能非常庞大。该大小取决于 DB2 配置和发生问题时进程的状态。

    如果设置了 **CORELIMIT**，那么 DB2 将使用此值来覆盖当前用户核心限制（ulimit）设置以生成核心文件。

    如果未设置 **CORELIMIT**，那么 DB2 会将核心文件大小设置为等于当前 ulimit 设置的值。

    **注**: 在下一次重新启动 DB2 实例之后，对用户核心限制或 **CORELIMIT** 所作的任何更改才会生效。

  **DUMPCORE**

  - 操作系统: Linux、Solaris 和 AIX
  - 缺省值: ON; 值: ON 或 OFF
  - 此选项指定是否将生成核心文件。核心文件是在 **diagpath** 目录中创建的用于问题确定的文件，它们包含终止 DB2 进程的整个进程映像。但是，实际是否转储核心文件取决于当前 ulimit 设置和 **CORELIMIT** 参数的值。某些操作系统还具有核心转储的配置设置，这些设置可能会规定应用程序核心转储的行为。

    建议通过将 **DUMPCORE** 设置为 OFF 来禁用核心文件转储。

  **DUMPDIR**

  - 操作系统: 所有操作系统
  - 缺省值: **diagpath** 目录，或者在未定义 **diagpath** 时为缺省诊断目录; 值: *目录的路径*
  - 此选项指定用于核心文件创建的目录的绝对路径名。

**DB2_FORCE_APP_ON_MAX_LOG**

- 操作系统: 所有操作系统
- 缺省值: TRUE; 值: TRUE 或 FALSE

- 指定当超过了 **max_log** 配置参数值时将发生的情况。如果设置为 TRUE，那么会强制应用程序断开与数据库的连接并回滚工作单元。

  如果设置为 FALSE，那么当前语句将失败。该应用程序仍然可以落实在工作单元中由先前语句完成的工作，它也可以回滚已完成的工作以撤销该工作单元。

  **注**：此 DB2 注册表变量影响 IMPORT 实用程序从日志满情况中恢复的能力。如果将 **DB2_FORCE_APP_ON_MAX_LOG** 设置为 TRUE，并且发出带 **COMMITCOUNT** 命令选项的 IMPORT 命令，那么 IMPORT 实用程序将不能执行落实以便避免用完活动日志空间。当 IMPORT 实用程序遇到 SQL0964C（事务日志满）时，将强制关闭数据库，并且当前工作单元将回滚。

**DB2GRAPHICUNICODESERVER**

- 操作系统：所有操作系统
- 缺省值：OFF；值：ON 或 OFF
- 此注册表变量用来存放现有的为将图形数据插入 Unicode 数据库而编写的应用程序。仅需要对那些特别地发送 Unicode 格式（而不是客户机的代码页）的 sqldbchar（图形）数据的应用程序使用此变量。（sqldbchar 是 C 和 C++ 中受支持的 SQL 数据类型，可以包含单个双字节字符。）如果设置为 ON，那么是通知数据库将接收的图形数据是 Unicode 格式，并且应用程序期望接收到 Unicode 格式的图形数据。

**DB2INCLUDE**

- 操作系统：所有操作系统
- 缺省值：当前目录
- 指定在 DB2 PREP 处理期间处理 SQL INCLUDE 文本文件语句时要使用的路径。它提供一个有可能在其中找到包含文件的目录列表。请参阅Developing Embedded SQL Applications以了解有关如何在不同预编译语言中使用 **DB2INCLUDE** 的描述。

**DB2INSTDEF**

- 操作系统：Windows
- 缺省值：DB2
- 此变量设置在未定义 **DB2INSTANCE** 时要使用的值。

**DB2INSTOWNER**

- 操作系统：Windows
- 缺省值：NULL
- 第一次创建实例时在 DB2 概要文件注册表中创建的注册表变量。将此变量设置为实例拥有的机器的名称。

**DB2_LIC_STAT_SIZE**

- 操作系统：所有操作系统
- 缺省值：NULL，范围：0 到 32767
- 此变量确定包含系统许可证统计信息的文件的最大大小（以 MB 计）。0 值关闭许可证统计信息收集。如果无法识别或未定义该变量，那么该变量缺省为不受限制的大小。使用许可证中心显示该统计信息。

**DB2LOCALE**

- 操作系统: 所有操作系统
- 缺省值: NO; 值: YES 或 NO
- 此变量指定在调用 DB2 之后是否将进程的缺省值“C”语言环境复原为缺省“C”语言环境，以及在调用 DB2 函数之后是否将进程语言环境复原为原始“C”。如果原始语言环境不是“C”，那么忽略此注册表变量。

**DB2_MAX_CLIENT_CONNRETRIES**

- 操作系统: 所有操作系统
- 缺省值: 未设置; 值: 重试连接的最大整数次数
- 此变量指定客户机自动重新路由功能尝试的最大连接重试次数。可以将此变量与 **DB2_CONNRETRIES_INTERVAL** 配合使用以配置客户机自动重新路由重试行为。

  如果设置了 **DB2_MAX_CLIENT_CONNRETRIES**，但未设置 **DB2_CONNRETRIES_INTERVAL**，那么 **DB2_CONNRETRIES_INTERVAL** 缺省为 30。如果未设置 **DB2_MAX_CLIENT_CONNRETRIES**，但设置了 **DB2_CONNRETRIES_INTERVAL**，那么 **DB2_MAX_CLIENT_CONNRETRIES** 缺省为 10。如果既未设置 **DB2_MAX_CLIENT_CONNRETRIES** 也未设置 **DB2_CONNRETRIES_INTERVAL**，那么客户机自动重新路由功能将恢复为采用缺省行为，即重复地重新尝试连接数据库，最多 10 分钟。

**DB2_OBJECT_TABLE_ENTRIES**

- 操作系统: 所有操作系统
- 缺省值: 0; 值: 0–65532

  系统上实际可能具有的最大值取决于页大小和扩展数据块大小，但是不能超过 65532。
- 此变量指定表空间中期望的对象数。如果知道将在 DMS 表空间中创建大量对象（例如，1000 或者更多个对象），那么在创建该表空间之前，应该将此注册表变量设置为一个接近的数目。在创建表空间期间，这将为对象元数据保留连续的存储器。保留连续的存储器会降低联机备份阻塞一些将更新元数据中的条目的操作（例如，CREATE INDEX 和 IMPORT REPLACE）的机率。它还将更容易调整表空间大小，原因是元数据将存储在表空间的开始部分。

  如果表空间的初始大小不足以保留连续存储器，那么将继续创建表空间而不保留附加空间。

**DB2_SYSTEM_MONITOR_ SETTINGS**

- 操作系统: 所有操作系统
- 此注册表变量控制一组参数，这些参数允许您修改 DB2 监视的各个方面的行为。用分号分隔每个参数，如以下示例所示:

  ```
  db2set DB2_SYSTEM_MONITOR_SETTINGS=OLD_CPU_USAGE:TRUE;
  DISABLE_CPU_USAGE:TRUE
  ```

每次设置 **DB2_SYSTEM_MONITOR_SETTINGS** 时，都必须显式设置每个参数。在设置此变量时未指定的任何参数将还原为其缺省值。因此，在以下示例中:

```
db2set DB2_SYSTEM_MONITOR_SETTINGS=DISABLE_CPU_USAGE:TRUE
```

**注**: 目前此注册表变量仅有针对 Linux 的设置，未来发行版将添加针对其他操作系统的其他设置。
OLD_CPU_USAGE 将复原为其缺省设置。

- 参数:

  **OLD_CPU_USAGE**

  - 操作系统: Linux
  - 值: TRUE/ON 或 FALSE/OFF
  - RHEL4 和 SLES9 上的缺省值: TRUE（注意，OLD_CPU_USAGE 的 FALSE 设置将被忽略 - 仅使用旧行为。）
  - RHEL5、SLES10 和其他操作系统上的缺省值: FALSE
  - 此参数控制实例如何获取 Linux 平台上的 CPU 使用时间。如果设置为 TRUE，那么将使用较旧的方法来获取 CPU 使用时间。此方法返回系统和用户 CPU 使用时间，但这样做将消耗更多 CPU（也就是说，它具有更高的开销）。如果设置为 FALSE，那么将使用较新的方法来获取 CPU 使用时间。此方法只返回用户 CPU 使用时间值，但由于它具有较小开销，所以速度更快。

  **DISABLE_CPU_USAGE**

  - 操作系统: Linux
  - 值: TRUE/ON 或 FALSE/OFF
  - RHEL4 和 SLES9 上的缺省值: TRUE
  - RHEL5、SLES10 和其他操作系统上的缺省值: FALSE
  - 此参数允许您确定是否读取 CPU 使用率。启用了 DISABLE_CPU_USAGE（设置为 TRUE）时，将不读取 CPU 使用率，从而避免在检索 CPU 使用率时有时会产生的开销。

**DB2TERRITORY**

- 操作系统: 所有操作系统
- 缺省值: 从语言标识中派生，由操作系统指定。
- 此变量指定客户机应用程序的区域或地域代码，它影响日期和时间格式。

**DB2_VIEW_REOPT_VALUES**

- 操作系统: 所有操作系统
- 缺省值: NO; 值: YES 或 NO
- 当说明重新优化的 SQL 或 XQuery 语句时，此变量使所有用户都能够将该语句的经过高速缓存的值存储在 EXPLAIN_PREDICATE 表中。当此变量设置为 NO 时，只允许 DBADM 将这些值保存在 EXPLAIN_PREDICATE 表中。

## 系统环境变量

### DB2_ALTERNATE_GROUP_LOOKUP

- 操作系统：AIX
- 缺省值：NULL；值：NULL 或 GETGRSET
- 此变量允许 DB2 数据库系统从操作系统提供的备用来源获取组信息。在 AIX 上，使用函数 getgrset。它使您能够通过可装入认证模块从本地文件以外的其他位置获取组。

### DB2_CLP_EDITOR

有关详细信息，请参阅“命令行变量”中的 DB2_CLP_EDITOR。

### DB2_CLP_HISTSIZE

有关详细信息，请参阅“命令行变量”中的 DB2_CLP_HISTSIZE。

### DB2CONNECT_ENABLE_EURO_CODEPAGE

- 操作系统：所有操作系统
- 缺省值：NO；值：YES 或 NO
- 在所有连接至 DB2 z/OS 版服务器或者 DB2 IBM i 版服务器（它们需要支持欧元符号）的 DB2 Connect 客户机和服务器中，将此变量设置为 YES。如果您将此变量设置为 YES，那么当前应用程序代码页将映射至显式指示对欧元符号的支持的等价编码字符集标识（CCSID）。因此，DB2 Connect 将使用一个 CCSID 来连接至 DB2 z/OS 版服务器或者 DB2 IBM i 版服务器，该 CCSID 是当前应用程序代码的 CCSID 的超集并且还支持欧元符号。例如，如果客户机使用的是映射至 CCSID 1252 的代码页，那么客户机将使用 CCSID 5348 来进行连接。

### DB2CONNECT_IN_APP_PROCESS

- 操作系统：所有操作系统
- 缺省值：YES；值：YES 或 NO
- 将此变量设置为 NO 时，将强制 DB2 Enterprise Server Edition机器上的本地 DB2 Connect 客户机在代理程序内运行。在代理程序内运行的一些优点是可监视本地客户机，且本地客户机可使用 SYSPLEX 支持。

### DB2_COPY_NAME

- 操作系统：Windows
- 缺省值：机器上安装的缺省 DB2 副本的名称。值：机器上安装的 DB2 副本的名称。此名称限长 128 个字符。
- **DB2_COPY_NAME** 变量存储当前正在使用的 DB2 副本的名称。如果机器上安装了多个 DB2 副本，且无法使用 **DB2_COPY_NAME** 来切换到另一个 DB2 副本，那么必须运行 *INSTALLPATH*\bin\db2envar.bat 命令来更改当前正在使用的副本，其中 *INSTALLPATH* 是 DB2 副本的安装位置。

### DB2DBMSADDR

- 操作系统：Linux on x86 和 Linux on zSeries（31 位）
- 缺省值：NULL；值：0x09000000 到 0xB0000000 范围内的虚拟地址，增量为 0x10000
- **DB2DBMSADDR** 注册表变量以十六进制格式指定缺省数据库共享内存地址。

**注**: 不正确的地址会导致 DB2 数据库系统产生严重问题，这些问题包括无法启动 DB2 实例，甚至无法连接至数据库。不正确的地址就是与内存中已经在使用的区域相冲突的地址或者是预先指定为用于其他某些用途的地址。要解决此问题，请通过使用以下命令将 **DB2DBMSADDR** 注册表变量复位为 NULL:

```
db2set DB2DBMSADDR=
```

此变量可用来精细调整 DB2 进程的地址空间布局。此变量将把实例共享内存的位置从它在虚拟地址 0x10000000 的当前位置更改为新值。

**DB2_DIAGPATH**

- 操作系统: 所有操作系统
- 缺省值: 在 UNIX 和 Linux 操作系统上，缺省值是实例 db2dump 目录；在 Windows 操作系统上，缺省值是实例 DB2 目录。
- 此参数仅适用于 ODBC 和 DB2 CLI 应用程序。

此参数允许您指定 DB2 诊断信息的标准路径。根据您使用的平台不同，此目录中可能包含转储文件、陷阱文件、错误日志、通知文件和警报日志文件。

对于该环境范围中的 ODBC 和 CLI 应用程序来说，设置此环境变量与设置 DB2 数据库管理器配置参数 **diagpath** 以及设置 CLI/ODBC 配置关键字 **DiagPath** 的效果相同。

**DB2DOMAINLIST**

- 操作系统: 所有操作系统
- 缺省值: NULL；值: Windows 域名列表，域名由逗号 (,) 分隔。
- 此变量定义一个或多个 Windows 域。此列表在服务器上维护，它定义针对其认证请求用户标识的域。只有属于这些域的用户会才会接受它们的连接或连接请求。

仅当在数据库管理器配置中设置了 CLIENT 认证时，此变量才有效。如果 Windows 域环境中要求从 Windows 桌面进行单点登录，那么需要此变量。

DB2 服务器版本 7.1 或更高版本支持 **DB2DOMAINLIST**，但仅在纯 Windows 域环境中。从版本 8 修订包 15 和版本 9.1 修订包 3 开始，如果客户机或服务器正在 Windows 环境中运行，那么将支持 **DB2DOMAINLIST**。

**DB2ENVLIST**

- 操作系统: UNIX
- 缺省值: NULL
- 此变量列示存储过程或用户定义的函数的特定变量名。缺省情况下，db2start 命令过滤掉除带 『DB2』 或 『db2』 前缀之外的所有用户环境变量。如果必须将特定环境变量传递至存储过程或用户定义的函数，那么可列示 **DB2ENVLIST** 环境变量中的变量名。用一个或多个空格将每个变量名隔开。

**DB2INSTANCE**

- 操作系统: 所有操作系统
- 缺省值: Windows 32 位操作系统上的 **DB2INSTDEF**。

- 此环境变量指定在缺省情况下处于活动状态的实例。在 UNIX 上，用户必须指定 **DB2INSTANCE** 的值。

  **注**: 无法使用 db2set 命令来更新此注册表变量。有关更多信息，请参阅第 339 页的『设置当前实例环境变量』和第 337 页的『在 Windows 上设置环境变量』。

**DB2INSTPROF**

- 操作系统: Windows
- 缺省值: `Documents and Settings\All Users\Application Data\IBM\DB2\`*`Copy Name`*（Windows XP 和 Windows 2003）或 `ProgramData\IBM\DB2\`*`Copy Name`*（Windows Vista）
- 此环境变量指定 Windows 操作系统上实例目录的位置。从版本 9.5 起，实例目录（和其他用户数据文件）不能在 `sqllib` 目录下。

**DB2LDAPSecurityConfig**

- 操作系统: 所有操作系统
- 缺省值: NULL; 值: IBM LDAP 安全插件配置文件的有效名称和路径
- 此变量用来指定 IBM LDAP 安全插件配置文件的位置。如果未设置此变量，那么 IBM LDAP 安全插件配置文件的名称为 `IBMLDAPSecurity.ini` 并且位于下列其中一个位置:
  - 在 Linux 和 UNIX 操作系统上: *`INSTHOME`*`/sqllib/cfg/`
  - 在 Windows 操作系统上: `%`**`DB2PATH`**`%\cfg\`

  在 Windows 操作系统上，应在全局系统环境中设置此变量，以确保 DB2 服务已使用此变量。

**DB2LIBPATH**

- 操作系统: UNIX
- 缺省值: NULL
- DB2 构造其自身的共享库路径。如果要将 PATH 添加到引擎的库路径中（例如，在 AIX 上，用户定义的函数在 **LIBPATH** 中需要特定的条目），那么必须设置 **DB2LIBPATH**。**DB2LIBPATH** 的实际值追加到 DB2 构造的共享库路径的末尾。

**DB2LOGINRESTRICTIONS**

- 操作系统: AIX
- 缺省值: LOCAL; 值: LOCAL、REMOTE、SU 或 NONE
- 此注册表变量允许您使用称为 loginrestrictions() 的 AIX 操作系统 API。此 API 确定是否允许用户访问系统。通过调用此 API，DB2 数据库安全性可以强制实施操作系统指定的登录限制。当使用此注册表变量时，可以向此 API 提交不同的值。这些值为:
  - REMOTE

    DB2 只强制实施登录限制来通过 rlogind 或 telnetd 程序验证帐户是否可用于远程登录。
  - SU

DB2 版本 9.1 只强制实施 su 限制来验证是否允许 su 命令，以及当前进程是否具有可通过调用 su 命令来切换到该帐户的组标识。

– NONE

DB2 不强制实施任何登录限制。

– LOCAL（或者不设置变量）

DB2 只强制实施登录限制来验证对于此帐户是否允许本地登录。这是登录时的正常行为。

不管您设置了哪一个选项，具有指定特权的用户帐户或标识都能够在服务器上以本地方式或者从远程客户机成功使用 DB2。有关 loginrestrictions() API 的描述，请参阅 AIX 文档。

**DB2NODE**

- 操作系统：所有操作系统
- 缺省值：NULL；值：1 到 999
- 用于指定希望连接的数据库分区服务器的目标逻辑节点。如果未设置此变量，目标逻辑节点将缺省为用该机器上的端口 0 定义的逻辑节点。在分区数据库环境中，连接设置可能会对获取可信连接产生影响。例如，如果 **DB2NODE** 变量设置为一个节点，以便在该节点上建立连接需要通过一个中间节点（中继段节点），那么在评估此连接以确定是否可以将它标记为可信连接时，将考虑该中间节点的 IP 地址和用于在中继段节点与连接节点之间进行通信的通信协议。也就是说，不考虑发出连接的原始节点。而是考虑中继段节点。

  **注**：无法使用 db2set 命令来更新此注册表变量。有关更多信息，请参阅第 337 页的『在 Windows 上设置环境变量』。

**DB2OPTIONS**

- 操作系统：所有操作系统
- 缺省值：NULL
- 用来设置命令行处理器选项。

**DB2_PARALLEL_IO**

- 操作系统：所有操作系统
- 缺省值：NULL；值：*TablespaceID*:[*n*],... - 用逗号分隔的已定义表空间（由其数字表空间标识表示）的列表。如果表空间的预取大小为 AUTOMATIC，那么可以通过指定表空间标识，后面跟一个冒号，再加上每个容器中的磁盘数 *n* 来向 DB2 数据库管理器表示该表空间的每个容器中的磁盘数。如果没有指定 *n*，那么将使用缺省值 6。

  您可以将 *TablespaceID* 替换为星号（*）来指定所有的表空间。例如，如果 **DB2_PARALLEL_IO** =*，那么所有表空间都将使用 6 作为每个容器中的磁盘数。如果您同时指定星号（*）和表空间标识，那么优先使用表空间标识设置。例如，如果 **DB2_PARALLEL_IO** =*,1:3，那么所有表空间都将使用 6 作为每个容器中的磁盘数，但第一个表空间除外（它使用 3 作为每个容器中的磁盘数）。

- 此注册表变量用来更改 DB2 计算表空间的 I/O 并行性的方式。当启用了 I/O 并行性时（要么通过使用多个容器来隐式启用，要么通过设置 **DB2_PARALLEL_IO** 来显式启用），通过发出正确数目的预取请求来实现此目标。每个预取请求都是对页的扩展数据块的请求。例如，表空间具有两个容器，而预取大小是扩展数据块大小的四倍。如果设置了注册表变量，那么此表空间的预取请求将分为四个请求（每个请求对应一个扩展数据块），并且可能由四个预取程序来并行处理这些请求。

  如果表空间中的各个容器分布在多个物理磁盘上，或者表空间中的容器是在由多个物理磁盘组成的单个 RAID 设备上创建的，那么您可能想要设置注册表变量。

  如果未设置此注册表变量，那么任何表空间的并行度都是表空间的容器数。例如，如果 **DB2_PARALLEL_IO** 设置为 NULL 并且表空间有四个容器，那么将发出四个按扩展数据块大小计算的预取请求；如果表空间有两个容器并且预取大小是扩展数据块大小的四倍，那么此表空间的预取请求将分为两个请求（每个请求对应两个扩展数据块）。

  如果设置了此注册表变量，并且表的预取大小不是 AUTOMATIC，那么表空间的并行度为预取大小除以扩展数据块大小。例如，如果对预取大小为 160 而扩展数据块大小为 32 页的表空间设置了 **DB2_PARALLEL_IO**，那么将发出五个按扩展数据块大小计算的预取请求。

  如果设置了此注册表变量，且表空间的预取大小是 AUTOMATIC，那么 DB2 将自动计算表空间的预取大小。下表总结了可用的不同选项和在每种情况下计算并行性的方式:

*表 63. 如何计算并行性*

| 表空间的预取大小 | DB2_PARALLEL_IO 设置 | 并行性等同于: |
|---|---|---|
| AUTOMATIC | 未设置 | 容器数 |
| AUTOMATIC | *表空间标识* | 容器数 * 6 |
| AUTOMATIC | *表空间标识*: *n* | 容器数 * *n* |
| Not AUTOMATIC | 未设置 | 容器数 |
| Not AUTOMATIC | *表空间标识* | 预取大小/扩展数据块大小 |
| Not AUTOMATIC | *表空间标识*: *n* | 预取大小/扩展数据块大小 |

  在某些情况下使用此变量可能导致磁盘争用。例如，如果表空间有两个容器，每个容器有专用的单个磁盘，那么设置此注册表变量可能导致这些磁盘发生争用，原因是两个预取程序将同时访问两个磁盘中的每个磁盘。但是，如果每个容器都分布在多个磁盘上，那么设置注册表变量将潜在允许同时访问四个不同的磁盘。

  要激活对此注册表变量的更改，请发出 db2stop 命令，然后输入 db2start 命令。

**DB2PATH**

- 操作系统: Windows

- 缺省值：随操作系统变化
- 此环境变量用于指定在 Windows 32 位操作系统上安装了该产品的目录。

**DB2PROCESSORS**

- 操作系统：Windows
- 缺省值：NULL；值：0–*n*-1（其中 *n*=处理器数）
- 此变量设置特定 db2syscs 进程的进程亲缘关系掩码。在运行多逻辑节点的环境中，这个变量用来将逻辑节点与处理器或一组处理器相关联。

  当指定了此变量时，DB2 将发出 SetProcessAffinityMask() API。如果未指定此变量，那么 db2syscs 进程与服务器上的所有处理器相关联。

**DB2RCMD_LEGACY_MODE**

- 操作系统：Windows
- 缺省值：NULL；值：YES、ON、TRUE 或 1，或者 NO、OFF、FALSE 或 0
- 此变量允许用户启用或禁用 DB2 远程命令服务的增强安全性功能。要以安全方式运行 DB2 远程命令服务，请将 **DB2RCMD_LEGACY_MODE** 设置为 NO、OFF、FALSE、0 或 NULL。要以旧方式（未增强安全性）运行，请将 **DB2RCMD_LEGACY_MODE** 设置为 YES、ON、TRUE 或 1。仅当域控制器运行的是 Windows 2000 或更高版本时，安全方式才可用。

  **注**：如果将 **DB2RCMD_LEGACY_MODE** 设置为 YES、ON、TRUE 或 1，那么所有发送到 DB2 远程命令服务的请求都在请求者的上下文中进行处理。为了便于进行此处理，必须在域控制器上启用机器和服务登录帐户，以允许机器和/或服务登录帐户模拟客户机。

  **注**：如果将 **DB2RCMD_LEGACY_MODE** 设置为 NO、OFF、FALSE 或 0，那么您必须拥有 SYSADM 权限才能让 DB2 远程命令服务代替您执行命令。

**DB2SYSTEM**

- 操作系统：Windows 和 UNIX
- 缺省值：NULL
- 指定您的用户和数据库管理员用于标识 DB2 数据库服务器系统的名称。应尽可能使此名称在网络中是唯一的。

  此名称显示在控制中心的对象树的系统层，以帮助管理员来标识可以从控制中心管理的服务器系统。

  当使用“配置助手”的搜索网络功能时，DB2 发现返回此名称，并在生成的对象树中的系统层显示该名称。此名称辅助用户标识包含他们希望访问的数据库的系统。在安装时，将 **DB2SYSTEM** 的值设置如下：
  - 在 Windows 上，安装程序将它设置成对 Windows 系统指定的计算机名称。
  - 在 UNIX 系统上，将它设置成 UNIX 系统的 TCP/IP 主机名。

**DB2_UPDDBCFG_SINGLE_DBPARTITION**

- 操作系统：所有操作系统
- 缺省值：未设置；值：0/FALSE/NO 或 1/TRUE/YES

- 当此注册表变量设置为 1、TRUE 或 YES 时，它允许您指定对数据库所进行的任何更新或复位仅影响特定分区。如果未设置此变量，那么更新和请求将按照版本 9.5 的行为。
- 从版本 9.5 开始，如果未指定分区子句，那么对数据库配置所进行的更新或更改将影响所有数据库分区。**DB2_UPDDBCFG_SINGLE_DBPARTITION** 允许您还原为先前版本的 DB2 的行为，即，对数据库配置的更新仅适用于本地数据库分区或 **DB2NODE** 注册表变量设置的数据库分区。对于需要此行为的任何现有命令脚本或应用程序，这提供了向后兼容性支持。

**注**：此变量不适用于通过调用 ADMIN_CMD 例程发出的更新或复位请求。

**DB2_USE_PAGE_CONTAINER_TAG**

- 操作系统：所有操作系统
- 缺省值：NULL；值：ON 或 NULL
- 缺省情况下，DB2 将容器标记存储在每个 DMS 容器的第一个扩展数据块中，而无论它是文件还是设备。容器标记是容器的元数据。在 DB2 V8.1 之前，容器标记存储在单个页中，因此，它在容器中需要较少的空间。要继续将容器标记存储在单个页中，请将 **DB2_USE_PAGE_CONTAINER_TAG** 设置为 ON。

  但是，如果在对容器使用 RAID 设备时将此注册表变量设置为 ON，那么会降低 I/O 性能。这是因为对于您使用等于 RAID 分割区大小或其倍数的扩展数据块大小创建表空间的 RAID 设备，将 **DB2_USE_PAGE_CONTAINER_TAG** 设置为 ON 会导致扩展数据块不与 RAID 分割区排列在一起。因此，I/O 请求可能需要访问比将优化的物理磁盘更多的磁盘。强烈建议用户启用此注册表变量，除非您有非常严格的空间约束，或者您要求行为与版本 8 之前的数据库行为保持一致。

  要激活对此注册表变量的更改，请发出 db2stop 命令，然后输入 db2start 命令。

**DB2_WORKLOAD**

- 操作系统：所有操作系统
- 缺省值：未设置；值：
  1C、CM、COGNOS_CS、FILENET_CM、INFOR_ERP_LN、MAXIMO、MDM、SAP、TPM、WAS、WC 或 WP
- **DB2_WORKLOAD** 的每个值表示若干个注册表变量和预定义的设置的特定分组。
- 以下是有效值:

  **1C** 当您想在数据库中为 1C 应用程序配置一组注册表变量时使用此设置。

  **CM** 当您想在数据库中为 IBM Content Manager 配置一组注册表变量时使用此设置。在 DB2 版本 9.5 修订包 3 和更高版本中提供此值。

  **COGNOS_CS**
  当您想在数据库中为 Cognos® Content Server 配置一组注册表变量时使用此设置。在 DB2 版本 9.5 修订包 5 和更高版本中提供了此值。

**FILENET_CM**

当您想在数据库中为 Filenet Content Manager 配置一组注册表变量时使用此设置。在 DB2 版本 9.5 修订包 5 和更高版本中提供了此值。

**INFOR_ERP_LN**

当您想在数据库中为 Infor ERP Baan 配置一组注册表变量时使用此设置。在 DB2 版本 9.5 修订包 6 和更高版本中提供了此值。

**MAXIMO**

当您想在数据库中为 Maximo® 配置一组注册表变量时使用此设置。在 DB2 版本 9.5 修订包 5 和更高版本中提供了此值。

**MDM** 当您想在数据库中为 Master Data Management（主数据管理）配置一组注册表变量时使用此设置。在 DB2 版本 9.5 修订包 5 和更高版本中提供了此值。

**SAP** 当您想在数据库中为 SAP 环境配置一组注册表变量时使用此设置。

设置了 **DB2_WORKLOAD**=SAP 时，不会自动创建用户表空间 SYSTOOLSPACE 和用户临时表空间 SYSTOOLSTMPSPACE。这些表空间用于由下列向导、实用程序或函数自动创建的表:

- 自动维护
- 设计顾问程序
- 控制中心数据库信息面板
- SYSINSTALLOBJECTS 存储过程（如果未指定表空间输入参数）
- GET_DBSIZE_INFO 存储过程

如果没有 SYSTOOLSPACE 和 SYSTOOLSTMPSPACE 表空间，那么不能使用这些向导、实用程序和函数。

为了能够使用这些向导、实用程序或函数，请执行下列任一操作:

- 手动创建 SYSTOOLSPACE 表空间以保存工具需要的对象（在分区数据库环境中，在目录分区上创建此表空间）。例如:

```
CREATE REGULAR TABLESPACE SYSTOOLSPACE
IN IBMCATGROUP
MANAGED BY SYSTEM
USING ('SYSTOOLSPACE')
```

- 通过指定有效的表空间，调用 SYSINSTALLOBJECTS 存储过程以创建用于工具的对象，并指定特定工具的标识。SYSINSTALLOBJECTS 将为您创建一个表空间。如果不想将 SYSTOOLSSPACE 用于对象，请指定另一个用户定义的表空间。

在至少完成了这些选项中的一个选项之后，创建 SYSTOOLSTMPSPACE 临时表空间（如果在分区数据库环境中工作，那么同样在目录分区上创建）。例如:

```
CREATE USER TEMPORARY TABLESPACE SYSTOOLSTMPSPACE
IN IBMCATGROUP
MANAGED BY SYSTEM
USING ('SYSTOOLSTMPSPACE')
```

创建了表空间 SYSTOOLSPACE 和临时表空间 SYSTOOLSTMPSPACE 之后，可以使用前面提及的向导、实用程序或函数。

**TPM** 当您想在数据库中为 IBM Tivoli Provisioning Manager 配置一组注册表变量时使用此设置。

**WAS** 当您想在数据库中为 WebSphere® Application Server 配置一组注册表变量时使用此设置。在 DB2 版本 9.5 修订包 5 和更高版本中提供了此值。

**WC** 当您想在数据库中为 WebSphere Commerce 配置一组注册表变量时使用此设置。在 DB2 版本 9.5 修订包 4 和更高版本中提供此值。

**WP** 当您想在数据库中为 WebSphere Portal 配置一组注册表变量时使用此设置。在 DB2 版本 9.5 修订包 5 和更高版本中提供了此值。

## 通信变量

### DB2CHECKCLIENTINTERVAL

- 操作系统: 所有操作系统，仅适用于服务器
- 缺省值 = 50; 值: 一个大于或等于零的数值。
- 此变量指定 TCP/IP 客户机连接验证的频率。它允许提前检测客户机的终止，而不是等到查询完成之后。如果此变量设置为 0，那么不执行任何验证。

  较低的值会使得检查频率更高。作为一条准则，对于较低频率，使用 100; 对于中等频率，使用 50; 对于较高频率，使用 10。使用 DB2 内部度量值来衡量此值。此值表示一个线性刻度，即将该值从 50 增加至 100 会使时间间隔加倍。当执行数据库请求时更频繁地检查客户机状态会延长完成查询所花的时间。如果 DB2 工作负载很重（即，它涉及许多内部请求），那么将 **DB2CHECKCLIENTINTERVAL** 设置为较低的值，较之于在工作负载较轻的情况对性能的影响会更大。

  从 DB2 Universal Database™版本 8.1.4 起，**DB2CHECKCLIENTINTERVAL** 的缺省值就为 50。在版本 8.1.4 之前，缺省值为 0。

### DB2COMM

- 操作系统: 所有操作系统，仅适用于服务器
- 缺省值 = NULL; 值: NPIPE、TCPIP 或 SSL
- 此变量指定当启动数据库管理器时所启动的通信管理器。如果未设置此变量，那么不在服务器上启动任何 DB2 通信管理器。

### DB2FCMCOMM

- 操作系统: 所有受支持的 DB2 Enterprise Server Edition平台
- 缺省值 = TCPIP4; 值: TCPIP4 或 TCPIP6
- 此变量指定如何解析 `db2nodes.cfg` 文件中的主机名。所有主机名都解析为 IPv4 或 IPv6。如果 `db2nodes.cfg` 中指定的是 IP 地址而不是主机名，那么 IP 地址的格式确定是使用 IPv4 还是 IPv6。如果未设置 **DB2FCMCOMM**，那么其缺省设置 IPv4 表示只能启动 IPv4 主机。

  **注**: 如果根据 `db2nodes.cfg` 中指定的主机名解析的 IP 格式或 `db2nodes.cfg` 中直接指定的 IP 格式与 **DB2FCMCOMM** 的设置不匹配，那么 db2start 将失败。

### DB2_FORCE_NLS_CACHE

- 操作系统: AIX、HP_UX 和 Solaris
- 缺省值 = FALSE; 值: TRUE 或 FALSE
- 此变量用于消去多线程应用程序中锁定争用的可能。当此注册表变量为 TRUE 时，在线程第一次访代码页和地域代码信息时将保存这些信息。由此，高速缓存的信息就可用于请求此信息的任何其他线程。在某些情况中，这样可消去锁定争用并导致有利于性能的结果。如果该应用程序更改了连接之间的语言环境设置，那么不应该使用此设置。在这种情况下，可能不需要这样做，原因是多线程应用程序通常不更改它们的语言环境设置，因为这样做不是线程安全的。

**DB2RSHCMD**

- 操作系统: UNIX
- 缺省值 = rsh（在 HP-UX 上为 remsh），值为 rsh、remsh 或 ssh 的完整路径名
- 缺省情况下，在启动远程数据库分区时，DB2 数据库系统使用 rsh 作为通信协议，并使用 db2_all 脚本来对所有数据库分区运行实用程序和命令。例如，将此注册表变量设置为 ssh 的完整路径名会导致 DB2 数据库产品使用 ssh 作为请求运行实用程序和命令的通信协议。还可以将它设置为使用适当缺省参数调用远程命令程序的脚本的完整路径名。仅分区数据库或以下单分区环境需要此变量: 在此环境中，db2start 命令是从安装了 DB2 产品的服务器之外的另一个服务器运行的。实例所有者必须能够使用指定的远程 shell 程序来从每个 DB2 数据库节点登录到其他每个 DB2 数据库节点，而不会提示他们输入任何其他验证或认证（即，密码或密码短语）。

有关设置 DB2RSHCMD 注册表变量以对 DB2 使用 ssh shell 的详细指示信息，请参阅“Configure DB2 Universal Database for UNIX to use OpenSSH”这本白皮书。

**DB2RSHTIMEOUT**

- 操作系统: UNIX
- 缺省值 = 30 秒; 值: 1 - 120
- 仅当 **DB2RSHCMD** 设置为非空值时，此变量才适用。此注册表变量用来控制 DB2 数据库系统将等待任何远程命令的超时时间段。经过这段超时时间段之后，如果未接收到响应，就会假定无法访问远程数据库分区，操作就会失败。

  **注**: 提供的时间值不是运行远程命令所需的时间，而是对请求进行认证所需的时间。

**DB2SORCVBUF**

- 操作系统: 所有操作系统
- 缺省值 = 65536
- 指定 TCP/IP 接收缓冲区的值。

**DB2SOSNDBUF**

- 操作系统: 所有操作系统
- 缺省值 = 65536
- 指定 TCP/IP 发送缓冲区的值。

**DB2TCP_CLIENT_CONTIMEOUT**

- 操作系统: 所有操作系统，仅适用于客户机
- 缺省值 = 0（无超时）；值: 0 - 32767 秒
- **DB2TCP_CLIENT_CONTIMEOUT** 注册表变量指定客户机等待 TCP/IP 连接操作完成的秒数。如果在指定的秒数内未建立连接，那么 DB2 数据库管理器将返回错误 `-30081 selectForConnectTimeout`。

  如果未设置此注册表变量或将它设置为 0，那么没有超时。

  **注**: 操作系统还有一个连接超时值，在达到您使用 **DB2TCP_CLIENT_CONTIMEOUT** 设置的超时之前起作用。例如，AIX 的缺省值为 *tcp_keepinit*=150（以 0.5 秒为单位），它将在 75 秒后终止连接。

**DB2TCP_CLIENT_RCVTIMEOUT**

- 操作系统: 所有操作系统，仅适用于客户机
- 缺省值 = 0（无超时）；值: 0 - 32767 秒
- **DB2TCP_CLIENT_RCVTIMEOUT** 注册表变量指定 TCP/IP 接收操作中客户机等待数据的秒数。如果在指定的秒数内未接收到来自服务器的数据，那么 DB2 数据库管理器将返回错误 `-30081 selectForRecvTimeout`。

  如果未设置此注册表变量或将它设置为 0，那么没有超时。

  **注**: CLI 可以使用 db2cli.ini 关键字 ReceiveTimeout 或连接属性 SQL_ATTR_RECEIVE_TIMEOUT 覆盖 DB2TCP_CLIENT_RCVTIMEOUT 的值。

**DB2TCPCONNMGRS**

- 操作系统: 所有操作系统
- 在串行机器上，缺省值 = 1；在对称多处理器上，缺省值为处理器数的平方根进位舍入到最大连接管理器数（16）。值: 1 至 16
- 如果未设置此注册表变量，那么创建缺省数目个连接管理器。如果设置了此注册表变量，那么此处指定的值覆盖缺省值。将创建指定数目个 TCP/IP 连接管理器，最多创建 16 个。如果指定小于 1 的值，那么 **DB2TCPCONNMGRS** 将设置为值 1，并记录一条警告，指示值超出范围。如果指定大于 16 的值，那么 **DB2TCPCONNMGRS** 将设置为值 16，并记录一条警告，指示值超出范围。1 到 16 之间的值按给定的那样使用。当创建多于 1 个连接管理器时，如果同时接收多个客户机连接，那么连接处理能力应能得到提高。如果用户在 SMP 机器上工作，或修改了 **DB2TCPCONNMGRS** 注册表变量，那么可能有其他 TCP/IP 连接管理器（在 UNIX 上）或线程（在 Windows 操作系统上）。附加的进程或线程需要附加的存储器。

  **注**: 将连接管理器数设置为 1 会导致具有许多用户的系统中的远程连接的性能降级、频繁连接和/或断开连接。

## 命令行变量

**DB2BQTIME**

- 操作系统: 所有操作系统

- 缺省值 = 1 秒，最小值: 1 秒
- 此变量指定命令行处理器前端在检查后端进程是否活动并建立与它的连接之前休眠的时间量。

**DB2BQTRY**

- 操作系统: 所有操作系统
- 缺省值 = 60 次重试，最小值: 0 次重试
- 此变量指定命令行处理器前端进程尝试确定后端进程是否活动的次数。此变量与 **DB2BQTIME** 共同起作用。

**DB2_CLP_EDITOR**

- 操作系统: 所有操作系统
- 缺省值: Notepad（Windows）或 vi（UNIX）; 值: 位于操作系统路径中的任何有效编辑器

  **注**: 在安装期间，不会将此注册表变量设置为缺省值。相反，如果未设置此注册表变量，使用此变量的代码就会使用缺省值。
- 此变量确定执行 EDIT 命令时要使用的编辑器。从 CLP 交互式会话中，EDIT 命令启动使用用户指定命令预装入的编辑器，于是就可编辑和运行该编辑器。

**DB2_CLP_HISTSIZE**

- 操作系统: 所有操作系统
- 缺省值: 20; 值: 1–500（包括 1 和 500）

  **注**: 在安装期间，不会将此注册表变量设置为缺省值。相反，如果未设置该注册表变量或者将它设置为有效范围之外的值，那么使用此变量的代码将使用缺省值 20。
- 此变量确定 CLP 交互式会话期间存储在命令历史记录中的命令数。由于命令历史记录保留在内存中，所以如果此变量的值非常高可能会对性能产生影响，影响大小取决于在会话中所运行命令的数目和长度。

**DB2_CLPPROMPT**

- 操作系统: 所有操作系统
- 缺省值=无（如果未定义它，那么“db2 =>”将用作缺省 CLP 交互提示符）; 值: 长度小于100且包含零个或多个下列标记的任何文本字符串: %i、%d、%ia、%da 或 %n。除非用户确实希望更改缺省 CLP 交互式提示符（db2 =>），否则他们不需要设置此变量。
- 此注册表变量允许用户定义要在“命令行处理器”（CLP）交互方式中使用的提示符。该变量可设置为长度小于100个字符且包含零个或多个可选标记 %i、%d、%ia、%da 或 %n 的任何文本字符串。当以 CLP 交互方式运行时，要使用的提示符的构造方式如下: 通过获取 **DB2_CLPPROMPT** 注册表变量中指定的文本字符串并将出现的所有 %i、%d、%ia、%da 或 %n 标记分别替换为当前连接的实例的本地别名、当前数据库连接的本地别名、当前连接的实例的授权标识、当前数据库连接的授权标识以及换行符（即，回车符）。

  **注**:

1. 如果在 CLP 交互方式下更改了 **DB2_CLPPROMPT** 注册表变量，那么在关闭并重新打开 CLP 交互方式之前，**DB2_CLPPROMPT** 的新值不会生效。
2. 如果不存在实例连接，那么 %ia 将替换为空字符串，而 %i 将替换为 **DB2INSTANCE** 注册表变量的值。（仅在 Windows 平台上）如果未设置 **DB2INSTANCE** 变量，那么 %i 将替换为 **DB2INSTDEF** 注册表变量的值。如果这些变量都没有设置，那么 %i 将替换为空字符串。
3. 如果不存在数据库连接，那么 %da 将替换为空字符串，而 %d 将替换为 **DB2DBDFT** 注册表变量的值。如果未设置 **DB2DBDFT** 变量，那么 %d 将替换为空字符串。
4. 交互式输入提示符将始终以大写形式显示授权标识、数据库名称和实例名的值。

**DB2IQTIME**

- 操作系统：所有操作系统
- 缺省值 = 5 秒，最小值：1 秒
- 此变量指定命令行处理器后端进程在输入队列上等待前端进程传送命令的时间量。

**DB2RQTIME**

- 操作系统：所有操作系统
- 缺省值 = 5 秒，最小值：1 秒
- 此变量指定命令行处理器后端进程等待前端进程提出请求的时间量。

## 分区数据库环境变量

**DB2CHGPWD_EEE**

- 操作系统：AIX、Linux 和 Windows 上的 DB2 ESE
- 缺省值 = NULL；值：YES 或 NO
- 此变量指定是否允许其他用户更改 AIX 或 Windows ESE 系统上的密码。必须确保在 Windows 上使用 Windows 域控制器或在 AIX 上使用 LDAP 来集中维护所有数据库分区或节点的密码。如果没有集中维护，在所有数据库分区或节点之间密码可能会不一致。这可能会导致只在用户为了更改而连接的数据库分区中更改密码。

**DB2_FCM_SETTINGS**

- 操作系统：Linux
- 缺省值 = NULL；值：
  - FCM_MAXIMIZE_SET_SIZE:[YES|TRUE|NO|FALSE]。FCM_MAXIMIZE_SET_SIZE 的缺省值为 NO。
  - FCM_CFG_BASE_AS_FLOOR: [YES|TRUE|NO|FALSE]。FCM_CFG_BASE_AS_FLOOR 的缺省值为 NO。
- 从版本 9.5 修订包 4 开始，可以设置 **DB2_FCM_SETTINGS** 注册表变量与 FCM_MAXIMIZE_SET_SIZE 选项，以便为快速通信管理器 (FCM) 缓冲区预先分配 4 GB 空间（缺省值）。要启用此功能，该标记的值必须为 YES 或 TRUE。

从版本 9.5 修订包 6 开始，可以设置 **DB2_FCM_SETTINGS** 注册表变量与 FCM_CFG_BASE_AS_FLOOR 选项，以将基本值设置为数据库管理器配置参数 *fcm_num_buffers* 和 *fcm_num_channels* 的最小值。当 FCM_CFG_BASE_AS_FLOOR 选项设置为 YES 或 TRUE，并且这些参数设置为 AUTOMATIC 且具有初始值或起始值时，DB2 不会将它们调整为低于此值。

**DB2_NUM_FAILOVER_NODES**

- 操作系统：所有操作系统
- 缺省值 = 2；值：0 至需要的数据库分区数
- 设置 **DB2_NUM_FAILOVER_NODES** 以指定在发生故障转移时可能需要在机器上启动的其他数据库分区的数目。

  在 DB2 数据库高可用性解决方案中，如果数据库服务器出现故障，那么可以在另一台机器上重新启动故障机器上的数据库分区。快速通信管理器（FCM）使用 **DB2_NUM_FAILOVER_NODES** 来计算每台机器上要保留以便于进行此故障转移的内存大小。

  例如，考虑以下配置：

  – 机器 A 有两个数据库分区：1 和 2。
  – 机器 B 有两个数据库分区：3 和 4。
  – **DB2_NUM_FAILOVER_NODES** 在机器 A 和机器 B 上都设置为 2。

  在 DB2START 时，FCM 将在机器 A 和机器 B 上保留足够多的内存，以便管理多达四个数据库分区，这样在一台机器出现故障时，可以在另一台机器上重新启动故障机器上的两个数据库分区。如果机器 A 出现故障，那么可以在机器 B 上重新启动数据库分区 1 和 2。如果机器 B 出现故障，那么可以在机器 A 上重新启动数据库分区 3 和 4。

**DB2_PARTITIONEDLOAD_DEFAULT**

- 操作系统：所有受支持的 ESE 平台
- 缺省值 = YES；值：YES 或 NO
- **DB2_PARTITIONEDLOAD_DEFAULT** 注册表变量允许用户在未指定特定于 ESE 的装入选项时更改 ESE 环境中 LOAD 实用程序的缺省行为。缺省值为 YES，这指定在 ESE 环境中，如果未指定特定于 ESE 的装入选项，那么将在定义了目标表的所有数据库分区上尝试装入。当值为 NO 时，仅在 LOAD 实用程序当前连接至的数据库分区上尝试装入。

  **注**：不推荐使用此变量，在以后的发行版中可能会将其除去。LOAD 命令具有各种选项，可使用它们获得相同的行为。通过使用 LOAD 命令指定以下内容，可获得与此变量的 NO 设置相同的结果：`PARTITIONED DB CONFIG MODE LOAD_ONLY OUTPUT_DBPARTNUMS x`，其中 x 是装入数据的分区的分区号。

**DB2PORTRANGE**

- 操作系统：Windows
- 值：nnnn:nnnn
- 将此值设置为 FCM 使用的 TCP/IP 端口范围，以便在另一台机器上创建的任何附加数据库分区也有同样的端口范围。

## 查询编译器变量

### DB2_ANTIJOIN

- 操作系统: 所有操作系统
- 缺省值 = NO（在 ESE 环境中）或 YES（在非 ESE 环境中）; 值: YES、NO 或 EXTEND
- 对于 DB2 Enterprise Server Edition: 当指定了 YES 时，优化器搜索将“NOT EXISTS”子查询变换为可以由 DB2 更有效处理的反连接的机会。对于非 ESE 环境: 当指定了 NO 时，优化器会限制将“NOT EXISTS”子查询变换为反连接的机会。

  在 ESE 和非 ESE 环境中，当指定了 EXTEND 时，优化器将搜索将“NOT IN”和“NOT EXISTS”子查询都变换为反连接的机会。

### DB2_INLIST_TO_NLJN

- 操作系统: 所有操作系统
- 缺省值 = NO; 值: YES 或 NO
- 在某些情况下，SQL 和 XQuery 编译器可以将 IN 列表谓词重写入连接。例如，在以下查询中:

```
SELECT *
  FROM EMPLOYEE         WHERE DEPTNO IN ('D11', 'D21', 'E21')
```

  可以编写为:

```
SELECT *
  FROM EMPLOYEE, (VALUES 'D11', 'D21', 'E21) AS V(DNO)
  WHERE DEPTNO = V.DNO
```

  如果 DEPTNO 上有索引，那么此修订版可以提供更好的性能。首先访问值列表，然后使用索引应用连接谓词，通过嵌套循环连接将它连接到 EMPLOYEE。

  有时，优化器没有准确的信息来确定用于查询的重写版本的最好连接方法。如果 IN 列表包含阻止优化器使用目录统计信息来确定选择性的参数标记或主变量，那么会发生这种情况。此注册表变量导致优化器使用表来支持嵌套循环连接来连接值列表，该表提供 IN 列表作为连接中的内部表。

  **注**: 当 DB2 查询编译器变量 **DB2_MINIMIZE_LISTPREFETCH** 和/或 **DB2_INLIST_TO_NLJN** 设置为 YES 时，它们都保持为活动状态，即使指定了 REOPT(ONCE) 亦如此。

### DB2_LIKE_VARCHAR

- 操作系统: 所有操作系统
- 缺省值 = Y,Y,
- 控制子元素统计信息的使用。当数据具有空格定界的一系列子字段或子元素形式的结构时，它们是列中具有关于数据内容的统计信息。子元素统计信息的收集是可选的并由 RUNSTATS 命令或 API 中的选项来控制。

  此注册表变量将影响优化器如何处理以下格式的谓词:

```
COLUMN LIKE '%xxxxxx%'
```

其中 xxxxxx 是任意一个字符串。

显示如何使用此注册表变量的语法如下:

```
db2set DB2_LIKE_VARCHAR=[Y|N|S|num1] [,Y|N|S|num2]
```

其中

– 逗号前的项或者谓词右边唯一的项，具有以下含义（仅当第二项指定为 N 或该列没有正的子元素统计信息）:
  - S - 优化器根据括在 % 字符中的字符串的长度，估计并置在一起构成一个列的一系列元素中每个元素的长度。
  - Y - 缺省值。对算法参数使用缺省值 1.9。将变长子元素算法与算法参数一起使用。
  - N - 使用定长子元素算法。
  - num1 - 将 num1 的值与变长子元素算法一起用作算法参数。

– 逗号后的项具有以下含义（仅适用于有正的子元素统计信息的列）:
  - N - 不要使用子元素统计信息。第一项生效
  - Y - 缺省值。使用变长子元素算法，该算法在列具有正的子元素统计信息的情况下，将子元素统计信息与算法参数的 1.9 缺省值结合使用。
  - num2 - 使用变长子元素算法，该算法在列具有正的子元素统计信息的情况下，将子元素统计信息与算法参数的 num2 的值结合使用。

### DB2_MINIMIZE_LISTPREFETCH

- 操作系统: 所有操作系统
- 缺省值 = NO; 值: YES 或 NO
- 列表预取是特殊的表访问方法，该方法涉及从索引检索限定 RID、按页号对其进行排序和预取数据页。有时，优化器没有准确的信息来确定列表预取是否是好的访问方法。当谓词选择性包含阻止优化器使用目录统计信息来确定选择性的参数标记或主变量时，可能发生这种情况。

  此注册表变量阻止优化器在这种情况下使用列表预取。

  **注**: 当 DB2 查询编译器变量 **DB2_MINIMIZE_LISTPREFETCH** 和/或 **DB2_INLIST_TO_NLJN** 设置为 YES 时，它们都保持为活动状态，即使指定了 REOPT(ONCE) 亦如此。

### DB2_NEW_CORR_SQ_FF

- 操作系统: 所有操作系统
- 缺省值 = OFF; 值: ON 或 OFF
- 将它设置为 ON 时，影响查询优化器为某些子查询谓词计算的选择值。它可用于改善等式子查询谓词的选择值的精确性，这些谓词使用该子查询的 SELECT 列表中的 MIN 或 MAX 聚集函数。例如:

```
SELECT * FROM T WHERE
T.COL = (SELECT MIN(T.COL)
FROM T WHERE ...)
```

### DB2_OPT_MAX_TEMP_SIZE

- 操作系统: 所有操作系统

- 缺省值 = NULL；值：一个查询在所有临时表空间中可以使用的空间大小（以兆字节计）
- 限制查询在临时表空间中可以使用的空间量。如果设置 **DB2_OPT_MAX_TEMP_SIZE**，那么优化器选择的方案有可能比其他方式下选择的方案成本高得多，但使用的临时表空间会较少。如果设置 **DB2_OPT_MAX_TEMP_SIZE**，请确保在限制临时表空间使用量的需要与由于此设置而选择的方案的效率之间进行权衡。

  如果设置了 **DB2_WORKLOAD=SAP**，那么将把 **DB2_OPT_MAX_TEMP_SIZE** 自动设置为 10240（10 GB）。

  如果运行的查询使用的临时表空间量超出对 **DB2_OPT_MAX_TEMP_SIZE** 设置的值，该查询不会失败，但是您将接收到一个警告，指出由于并非所有资源都可用，所以该查询的性能可能达不到最佳效果。

  优化器考虑的受 **DB2_OPT_MAX_TEMP_SIZE** 设置的限制影响的操作是：
  - 诸如 ORDER BY、DISTINCT、GROUP BY、合并扫描连接和嵌套循环连接之类的操作的显式排序。
  - 显式临时表
  - 散列连接和双重合并连接的隐式临时表

**DB2_REDUCED_OPTIMIZATION**

- 操作系统：所有操作系统
- 缺省值 = NO；值：NO、YES、任何整数、DISABLE、NO_SORT_NLJOIN 或 NO_SORT_MGJOIN
- 此注册表变量允许您请求使用缩减的优化功能或严格按照指定的优化类来使用优化功能。如果减少使用优化技术的次数，也会减少优化期间时间和资源的使用。

  **注：** 尽管可以减少优化时间和资源的使用，但会增大产生小于最佳数据访问方案的风险。仅当 IBM 或其合作伙伴建议时，才使用此注册表变量。
  - 如果设置为 NO

    优化器将不更改它的优化技术。
  - 如果设置为 YES

    如果优化类为 5（缺省值）或更低，优化器会禁用某些优化技术，这些技术可能消耗显著的准备时间和资源，但通常不会产生更好的访问方案。

    如果优化类恰好是 5，优化器按比例逐步减少或禁用某些附加技术，这些技术可能进一步减少优化时间和资源使用，但也进一步增加小于最佳访问方案的风险。对于小于 5 的优化类，其中某些技术不可能在任何情况下都有效。但是，如果它们有效，那么它们仍然有效。
  - 如果设置为任何整数

    效果与 YES 相同，对于在级别 5 优化的动态准备的查询，具有以下附加行为。如果任何查询块中的连接总数超出设置，那么优化器切换至贪婪连接枚举，而不是禁用附加的优化技术，如以上对级别 5 优化类所述。这意味将以与优化类 2 类似的级别优化查询。

– 如果设置为 DISABLE

当不受此 **DB2_REDUCED_OPTIMIZATION** 变量约束时，优化器的行为有时会对优化类 5 的动态查询动态地减少优化，此设置禁用此行为并要求优化器执行全部的级别 5 优化。

– 如果设置为 NO_SORT_NLJOIN

优化器不生成对嵌套循环连接（NLJN）强制排序的查询方案。这些类型的排序有利于提高性能；因此，使用 NO_SORT_NLJOIN 选项时务必谨慎，它可能会对性能产生严重影响。

– 如果设置为 NO_SORT_MGJOIN

优化器不生成对合并扫描连接（MSJN）强制排序的查询方案。这些类型的排序有利于提高性能；因此，使用 NO_SORT_MGJOIN 选项时务必谨慎，它可能会对性能产生严重影响。

注意：优化类 5 的动态优化减少优先于对于在 **DB2_REDUCED_OPTIMIZATION** 设置为 YES 时恰好是 5 的优化类所描述的行为以及整数设置所描述的行为。

### DB2_SELECTIVITY

- 操作系统：所有操作系统
- 缺省值 = NO；值：YES 或 NO
- 此注册表变量控制在 SQL 语句的搜索条件中可以使用 SELECTIVITY 子句的情况。

  当此注册表变量设置为 YES 时，可以为下列谓词指定 SELECTIVITY 子句：
  – 最少有一个表达式包含主变量的基本谓词
  – 其中的 MATCH 表达式、谓词表达式或转义表达式包含主变量的 LIKE 谓词

### DB2_SQLROUTINE_PREPOPTS

- 操作系统：所有操作系统
- 缺省值 = 空字符串；值：
  – BLOCKING {UNAMBIG | ALL | NO}
  – DATETIME {DEF | USA | EUR | ISO | JIS | LOC}
  – DEGREE {1 | *degree-of-parallelism* | ANY}
  – DYNAMICRULES {BIND | INVOKEBIND | DEFINEBIND | RUN | INVOKERUN | DEFINERUN}
  – EXPLAIN {NO | YES | ALL}
  – EXPLSNAP {NO | YES | ALL}
  – FEDERATED {NO | YES}
  – INSERT {DEF | BUF}
  – ISOLATION {CS | RR | UR | RS | NC}
  – OPTPROFILE {profile_name | schema_name.profile_name}
  – QUERYOPT *optimization-level*

– REOPT {NONE | ONCE | ALWAYS}

– VALIDATE {RUN | BIND}

- **DB2_SQLROUTINE_PREPOPTS** 注册表变量可以用来定制 SQL 和 XQuery 过程的预编译和绑定选项。设置此变量时，请使用空格分隔每个选项，如下所示:

```
db2set DB2_SQLROUTINE_PREPOPTS="BLOCKING ALL VALIDATE RUN"
```

有关每个选项及其设置的完整描述，请参阅“BIND 命令”。

如果想要对选择各个过程获得与 **DB2_SQLROUTINE_PREPOPTS** 相同的结果，但不需要重新启动实例，那么可使用 SET_ROUTINE_OPTS 过程。

## 性能变量

### DB2_ALLOCATION_SIZE

- 操作系统: 所有操作系统
- 缺省值: 128 KB，范围: 64 KB - 256 MB
- 指定缓冲池的内存分配大小。

  为此注册表变量设置较高的值可能具有下列优点: 将需要更少的分配以达到缓冲池的期望内存量。

  为此注册表变量设置较高的值可能会增加成本，这是因为如果缓冲池的更改量不是分配大小的倍数，那么会浪费内存。例如，如果 **DB2_ALLOCATION_SIZE** 的值为 8 MB，而缓冲池减少了 4 MB，那么由于不能释放整个 8 MB 段，所以这 4 MB 将会浪费。

**注**: 不推荐使用 **DB2_ALLOCATION_SIZE**，在以后的发行版中可能会将其除去。

### DB2_APM_PERFORMANCE

- 操作系统: 所有操作系统
- 缺省值: OFF; 值: ON 或 OFF
- 将此变量设置为 ON 来启用访问方案管理器 (APM) 中与性能相关的更改，这些更改将影响查询高速缓存（程序包高速缓存）的行为。通常建议不要对生产系统使用这些设置。它们会引入一些限制，例如，可能会出现包外高速缓存错误和/或增加内存使用量。

  将 **DB2_APM_PERFORMANCE** 设置为 ON 也会启用无包锁定方式。此方式允许全局查询高速缓存运行，而不必使用包锁定，这些锁定是内部系统锁定，可以保护高速缓存的包条目不会被除去。无包锁定方式在一定程度上可以提高性能，但它不允许执行某些数据库操作。这些被禁止的操作可能包括: 使包无效的操作、使包不起作用的操作、PRECOMPILE、BIND 和 REBIND。

### DB2ASSUMEUPDATE

- 操作系统: 所有操作系统
- 缺省值: OFF; 值: ON 或 OFF

- 当启用此变量时，它允许 DB2 数据库系统假定正在更改 UPDATE 语句中提供的所有定长列。这使得 DB2 数据库系统无需将现有列值与新值进行比较就可以确定实际上是否正在进行更改。当为更新提供了一些列（例如，在 SET 子句中）但是实际上没有进行修改时，使用此注册表变量会导致更多日志记录和索引维护。

  在执行 db2start 命令时激活 **DB2ASSUMEUPDATE** 注册表变量有效。

**DB2_ASYNC_IO_MAXFILOP**

- 操作系统：所有操作系统
- 缺省值：**maxfilop** 配置参数的值；值：从 **maxfilop** 的值到 **max_int** 的值
- 不推荐使用 **DB2_ASYNC_IO_MAXFILOP**，在以后的发行版中可能会将其除去。由于线程数据库管理器所维护的共享文件句柄表而使此变量已过时。有关更多信息，请参阅*《数据服务器、数据库和数据库对象指南》*中的“共享文件句柄表”。

  在版本 9.5 中仍然可以设置 **DB2_ASYNC_IO_MAXFILOP**，但是它将不起作用。如果您想限制可以为每个数据库打开的文件句柄数，那么使用 **maxfilop** 配置参数。

**DB2_AVOID_PREFETCH**

- 操作系统：所有操作系统
- 缺省值：OFF；值：ON 或 OFF
- 指定在崩溃恢复期间是否应使用预取。如果 **DB2_AVOID_PREFETCH** =ON，那么不使用预取。

**DB2BPVARS**

- 操作系统：对每个参数指定的操作系统
- 缺省值：路径
- 两组参数可用于调整缓冲池。一组参数（仅在 Windows 上可用）指定缓冲池应对特定类型的容器使用散射读。另一组参数（在所有平台上都可用）影响预取行为。

  以格式 `parameter=value` 在 ASCII 码文件中指定参数，每行一个参数。例如，名为 `bpvars.vars` 的文件可能包含下列行:

  ```
  NO_NT_SCATTER = 1
  NUMPREFETCHQUEUES = 2
  ```

  假定 `bpvars.vars` 存储在 `F:\vars\` 中，可执行下列命令来设置这些变量:

  ```
  db2set DB2BPVARS=F:\vars\bpvars.vars
  ```

  **散射读参数**

  对于针对相应类型的容器有大量顺序预取的系统以及您已将 **DB2NTNOCACHE** 设置为 ON 的系统，建议使用这些散射读参数。这些参数仅在 Windows 平台上可用，它们是 **NT_SCATTER_DMSFILE**、**NT_SCATTER_DMSDEVICE** 和 **NT_SCATTER_SMS**。指定 **NO_NT_SCATTER** 参数以显式地禁止对任何容器进行散射读。特定参数用

于对所指示类型的所有容器打开散射读。对于这些参数中的每个参数，缺省值为零（或 OFF）；且可能的值包括：零（或 OFF）和 1（或 ON）。

**注**：仅当 **DB2NTNOCACHE** 设置为 ON 来关闭 Windows 文件高速缓存时，才可以打开散射读。如果 **DB2NTNOCACHE** 设置为 OFF 或者未设置，那么当您试图对任何容器打开散射读时，会向管理通知记录中写入一条警告消息，且仍然禁用散射读。

**预取调整参数**

预取调整参数是 **NUMPREFETCHQUEUES** 和 **PREFETCHQUEUESIZE**。这些参数在所有平台上都可用，且可用于改进缓冲池数据预取。例如，请考虑期望的 **PREFETCHSIZE** 划分为 **PREFETCHSIZE/EXTENTSIZE** 预取请求的顺序预取。在这种情况下，将请求放置在预取队列上，从该队列分派 I/O 服务器来执行异步 I/O。缺省情况下，DB2 数据库管理器为每个数据库分区维护一个大小为 max(200,2***NUM_IOSERVERS**) 的队列。在某些环境中，随着队列和/或不同大小的队列的增加，性能会得到提高。预取队列的数目应该最多为 I/O 服务器数目的一半。当设置这些参数时，考虑其他参数（如 **PREFETCHSIZE**、**EXTENTSIZE**、**NUM_IOSERVERS** 和缓冲池大小）以及工作负载特征（如当前用户数）。

如果认为缺省值对于您的环境太小，首先稍稍增大该值。例如，您可以设置 **NUMPREFETCHQUEUES**=4 和 **PREFETCHQUEUESIZE**=200。以受控方式对这些参数进行更改，以便可以监视和评估这些更改的效果。

对于 **NUMPREFETCHQUEUES**，缺省值为 1，该值的范围是 1 至 **NUM_IOSERVERS**。如果将 **NUMPREFETCHQUEUES** 设置为小于 1，那么将它调整为 1。如果将它设置为大于 **NUM_IOSERVERS**，那么将它调整为 **NUM_IOSERVERS**。

对于 **PREFETCHQUEUESIZE**，缺省值为 max(200,2***NUM_IOSERVERS**)。值的范围是 1 到 32767。如果将 **PREFETCHQUEUESIZE** 设置为小于 1，那么将它调整为缺省值。如果设置为大于 32767，那么将它调整为 32767。

**注**：建议您不要使用 **DB2BPVARS**，在以后的发行版中可能会将其除去。

**DB2CHKPTR**

- 操作系统：所有操作系统
- 缺省值：OFF；值：ON 或 OFF
- 指定是否需要对输入进行指针检查。

**DB2CHKSQLDA**

- 操作系统：所有操作系统
- 缺省值：ON；值：ON 或 OFF
- 指定是否需要对输入进行 SQLDA 检查。

**DB2_EVALUNCOMMITTED**

- 操作系统：所有操作系统
- 缺省值：NO；值：YES 或 NO

- 当启用此变量时，在可能的情况下，它将进行表或索引访问扫描以延迟或避免行锁定，直到知道数据记录满足谓词求值为止。

  当启用此变量时，可对未落实的数据进行谓词求值。

  **DB2_EVALUNCOMMITTED** 只适用于使用“游标稳定性”或“读稳定性”隔离级别的语句。对于索引扫描，索引必须是 2 类索引。

  而且，除非还设置了注册表变量 **DB2_SKIPDELETED**，否则，对于表扫描访问将无条件地跳过已删除的行，而对于 2 类索引扫描则不会跳过已删除的键。

  在执行 db2start 命令时激活 **DB2_EVALUNCOMMITTED** 注册表变量有效。有关延迟锁定是否合适的决定则是在语句编译或绑定时作出的。

**DB2_EXTENDED_IO_FEATURES**

- 操作系统: AIX
- 缺省值: OFF; 值: ON 或 OFF
- 将此变量设置为 ON 可启用增强 I/O 性能的功能。此增强功能包括提高内存高速缓存的命中率以及减少优先级高的 I/O 的等待时间。这些功能仅可用于特定的软硬件配置组合；对于其他配置来说，将此变量设置为 ON 将被 DB2 数据库管理系统或操作系统忽略。最低配置要求为:
  - 数据库版本: DB2 V9.1
  - 必须使用原始设备作为数据库容器（文件系统上的容器不受支持）
  - 操作系统: AIX 5.3 TL4
  - 存储器子系统: Shark DS8000® 支持所有增强的 I/O 性能特征。请参阅 Shark DS8000 文档以了解设置和先决条件信息。

  缺省 HIGH、MEDIUM 和 LOW I/O 优先级设置分别为 3、8 和 12; 您可以使用 **DB2_IO_PRIORITY_SETTING** 注册表变量来更改这些设置。

**DB2_EXTENDED_OPTIMIZATION**

- 操作系统: 所有操作系统
- 缺省值: OFF; 值: ON、OFF、ENHANCED_MULTIPLE_DISTINCT 或 IXOR
- 此变量指定查询优化器是否使用优化扩充功能来提高查询性能。ON 和 ENHANCED_MULTIPLE_DISTINCT 值指定不同的优化扩充功能。如果要使用这两个值，请使用用逗号分隔的列表。

  如果查询中的一个单选操作涉及多个不同的聚集操作并且处理器数与数据库分区数的比率很低（例如，比率小于或等于 1），那么可以使用 ENHANCED_MULTIPLE_DISTINCT 值来提高这种查询的性能。此设置应该用在未使用对称多处理器（SMP）的 DPF（数据库分区功能）环境中。

  IXOR 选项（从 DB2 版本 9.5 修订包 6 发行版开始提供此选项）指定优化器将使用索引 OR 运算数据访问方法。如果 **DB2_WORKLOAD** 设置为 SAP，那么会自动将 **DB2_EXTENDED_OPTIMIZATION** 设置为 IXOR。

  优化扩充功能并不能在所有环境中提高查询性能。应执行测试来确定个别的查询性能提高。

**DB2_HASH_JOIN**

- 操作系统: 所有操作系统
- 缺省值: YES; 值: YES 或 NO
- 在编译访问方案时，指定散列连接作为可能的连接方法。需要调整 **DB2_HASH_JOIN** 以获得最佳性能。如果能够避免散列循环以及溢出到磁盘，那么散列连接的性能最佳。要调整散列连接性能，估计对 **sheapthres** 配置参数可用的最大内存量，然后调整 **sortheap** 配置参数。增大它的值，直到尽可能避免散列循环和磁盘溢出，但不要达到由 **sheapthres** 配置参数指定的限制。

  **注**: 在版本 9.5 中，不推荐使用 **DB2_HASH_JOIN**，将来的发行版中可能会将它除去。

**DB2_IO_PRIORITY_SETTING**

- 操作系统: AIX
- 值: HIGH: *#*, MEDIUM: *#*, LOW: *#*, 其中 # 可以是 1 至 15
- 此变量和 **DB2_EXTENDED_IO_FEATURES** 注册表变量一起使用。此注册表变量提供了一种方式来覆盖 DB2 数据库系统的缺省 HIGH、MEDIUM 和 LOW I/O 优先级设置，该缺省设置分别为 3、8 和 12。此注册表变量必须在启动实例前进行设置，修改此变量要求重新启动实例。注意，仅设置此注册表变量不会启用增强的 I/O 功能，必须再设置 **DB2_EXTENDED_IO_FEATURES** 才能启用这些功能。**DB2_EXTENDED_IO_FEATURES** 的所有系统要求同样适用于此注册表变量。

**DB2_KEEP_AS_AND_DMS_ CONTAINERS_OPEN**

- 操作系统: 所有操作系统
- 缺省值: NO; 值: YES 或 NO
- 将此变量设置为 ON 时，每个 DMS 表空间容器都会打开一个文件句柄，直到数据库被取消激活。因为去掉了打开容器的开销，所以查询性能可能会提高。仅应在纯 DMS 环境中使用此注册表，否则针对 SMS 表空间的查询的性能可能会受到负面影响。

**DB2_KEEPTABLELOCK**

- 操作系统: 所有操作系统
- 缺省值: OFF; 值: ON、TRANSACTION、OFF 或 CONNECTION
- 如果关闭了“未落实的读”或“游标稳定性”隔离级别，那么当此变量设置为 ON 或 TRANSACTION 时，它允许 DB2 数据库系统保持表锁定。当事务结束时将释放保持的表锁定，就像为“读稳定性”和“可重复读”扫描释放表锁定一样。

  当此变量设置为 CONNECTION 时，将对应用程序释放表锁定，直到应用程序回滚事务或连接复位。在落实之间，表锁定将继续挂起，删除表锁定的应用程序请求将被数据库忽略。表锁定仍然被分配给该应用程序。因此，当应用程序重新请求表锁定时，锁定已经处于可用状态。

  对于可利用此优化的应用程序工作负载，性能应该会有所提高。但是，并行执行的其他应用程序的工作负载可能会受到影响。其他应用程序可能无法访

问给定表，从而使得并行性过低。DB2 SQL 目录表不受此设置影响。CONNECTION 设置还包括使用 ON 或 TRANSACTION 设置描述的行为。

在进行语句编译或绑定时将检查此注册表变量。

**DB2_LARGE_PAGE_MEM**

- 操作系统：AIX、Linux 和 Windows Server 2003
- 缺省值：NULL，值：使用 * 来表示应使用大内存页内存的所有适用内存区域或者应使用大内存页内存的特定内存区域列表（各内存区域之间用逗号分隔）。可用区域根据操作系统有所变化。在 AIX 上，可以指定下列区域：DB、DBMS、FCM、APPL 或 PRIVATE。在 Linux 上，可以指定以下区域：DB。在 Windows Server 2003 上，可以指定以下区域：DB。巨页内存仅在 AIX 上可用。
- **DB2_LARGE_PAGE_MEM** 注册表变量用来启用大内存页支持或者巨页支持。通过设置 **DB2_LARGE_PAGE_MEM**=DB，就可以对数据库共享内存区域启用大内存页内存；如果 **database_memory** 设置为 AUTOMATIC，那么将禁止由 STMM 自动调整此共享内存区域。在 AIX 上，通过设置 DB2_LARGE_PAGE_MEM=DB:16GB，就可以对数据库共享内存区域启用巨页内存。

  使用大量虚拟内存的密集内存访问型应用程序可以通过使用大内存页或巨页来提高性能。为了使 DB2 数据库系统可以使用大内存页或巨页，必须先将操作系统配置为使用大内存页或巨页。

  要对 64 位 DB2 AIX 版上的代理程序专用内存启用大内存页（**DB2_LARGE_PAGE_MEM**=PRIVATE 设置），您必须在操作系统上配置大内存页，并且实例所有者必须具有 CAP_BYPASS_RAC_VMM 和 CAP_PROPAGATE 能力。

  在 AIX 5L™ 上，可以将此变量设置为 FCM。FCM 内存位于它自己的内存集中，因此必须将 FCM 关键字添加至 **DB2_LARGE_PAGE_MEM** 注册表变量，以便对 FCM 内存启用大内存页。

  在 Linux 上，还要求 `libcap.so.1` 库可用。必须先安装此库才能使此选项起作用。如果此选项已打开，且该库不在系统上，那么 DB2 数据库会禁用大内核页并继续如常运作。

  在 Linux 上，要验证大内核页是否可用，请发出以下命令：

  ```
  cat /proc/meminfo
  ```

  如果大内核页可用，那么应该出现下面三行（服务器上配置的内存量不同，显示的数目也会不同）：

  ```
  HugePages_Total:   200
     HugePages_Free:    200
     Hugepagesize:    16384 kB
  ```

  如果没看到这几行，或者如果 `HugePages_Total` 为 0，那么需要配置操作系统或内核。

  在 Windows 上，系统上可用的大内存页内存量小于总可用内存。系统运行一段时间之后，内存就会被分段，大页内存量就会减少。

**DB2_LOGGER_NON_ BUFFERED_IO**

- 操作系统: 所有操作系统
- 缺省值: OFF; 值: ON 或 OFF
- 此变量对日志文件系统启用直接 I/O。

从 DB2 版本 9.5 修订包 1 发行版开始提供 **DB2_LOGGER_NON_BUFFERED_IO** 注册表变量。

**DB2MAXFSCRSEARCH**

- 操作系统: 所有操作系统
- 缺省值: 5; 值: -1、1 至 33554
- 指定在将记录添加至表中时, 要搜索的可用空间控制记录 (FSCR) 数。缺省情况是搜索 5 个 FSCR。修改此值允许您平衡插入速度与空间复用。使用较大的值将优化空间复用。使用较小的值将优化插入速度。将值设置为 -1 会强制数据库管理器搜索所有 FSCR。

**DB2_MAX_INACT_STMTS**

- 操作系统: 所有操作系统
- 缺省值: 未设置; 值: 最大为 4 000 000 000
- 此变量覆盖对任何一个应用程序保留的不活动语句数的缺省限制。可以选择另一个值, 以便增大或减小用于不活动语句信息的系统监视器堆大小。缺省限制为 250。

  如果应用程序在一个工作单元中包含数目极多的语句, 或者如果同时有大量应用程序正在执行, 那么可能会耗尽系统监视器堆。

**DB2_MAX_NON_TABLE_LOCKS**

- 操作系统: 所有操作系统
- 缺省值: YES; 值: 请参阅描述
- 此变量定义在事务释放所有 NON 表锁定之前该事务可具有的 NON 表锁定的最大数目。NON 表锁定是这样一些表锁定: 即使事务已使用完它们, 它们仍保留在散列表和事务链中。因为事务经常多次访问同一个表, 所以保留锁定并将它们的状态更改为 NON 可以提高性能。

  为了获得最佳结果, 此变量的建议值是想要任何连接访问的表的最大数目。如果未指定用户定义的值, 那么缺省值如下所示: 如果**锁定列表**大小大于或等于

  ```
  SQLP_THRESHOLD_VAL_OF_LRG_LOCKLIST_SZ_FOR_MAX_NON_LOCKS
  ```

  (当前为 8000), 缺省值是

  ```
  SQLP_DEFAULT_MAX_NON_TABLE_LOCKS_LARGE
  ```

  (当前为 150)。否则, 缺省值为

  ```
  SQLP_DEFAULT_MAX_NON_TABLE_LOCKS_SMALL
  ```

  (当前为 0)。

**DB2_MDC_ROLLOUT**

- 操作系统: 所有操作系统

- 缺省值: IMMEDIATE; 值: IMMEDIATE、OFF 或 DEFER
- 此变量对 MDC 表的删除操作启用称为“转出”的性能增强功能。当在搜索式 DELETE 语句中删除整个单元格（维值的交叉点）时，转出是删除 MDC 表中的行的较快方法。好处是减少了记录工作而处理效率更高。
- 变量设置有三种可能的结果:
  - 不转出 - 在指定了 OFF 时
  - 立即转出 - 在指定了 IMMEDIATE 时
  - 转出但延迟清除索引 - 在指定了 DEFER 时
- 如果在启动后更改了值，那么对语句进行的任何新编译都将沿用新的注册表值设置。对于程序包高速缓存中的语句，在重新编译语句之前，不会在删除处理中进行任何更改。SET CURRENT MDC ROLLOUT MODE 语句将在应用程序连接级别覆盖 **DB2_MDC_ROLLOUT** 的值。

**DB2MEMDISCLAIM**

- 操作系统: 所有操作系统
- 缺省值: YES; 值: YES 或 NO
- DB2 数据库系统进程使用的内存可能有一些关联的调页空间。即使在已经释放关联的内存后，也会保留该调页空间。是否如此，取决于操作系统的（可调）虚拟内存管理资源分配策略。**DB2MEMDISCLAIM** 注册表变量控制 DB2 代理程序是否显式请求操作系统从释放的内存中解除关联保留的调页空间。

  **DB2MEMDISCLAIM** 设置为 YES 会导致较少的调页空间需求，并且可能导致较少的页面调度磁盘活动。**DB2MEMDISCLAIM** 设置为 NO 将会导致较多的调页空间需求，并且可能导致更多的页面调度磁盘活动。在某些情况下，例如，如果调页空间很大，并且实内存很大从不会发生页面调度，那么设置为 NO 会提供较小的性能提高。

**DB2MEMMAXFREE**

- 操作系统: 所有操作系统
- 缺省值: NULL; 值: 0 至 $2^{32}$-1 字节
- 指定未使用的专用内存的最大字节数，在将未使用的内存返回给操作系统之前这些内存由 DB2 数据库系统进程保留。

  如果未设置 **DB2MEMMAXFREE**，那么在将内存释放回操作系统之前，DB2 数据库系统进程将保留多达 20% 的未使用专用内存（根据当前消耗的专用内存量）。

**注**: 建议您不要使用 **DB2MEMMAXFREE**，在以后的发行版中会将其除去。由于数据库管理器现在使用线程引擎模型，所以不再需要此变量。不要设置此变量。设置它可能会降低性能，并且可能导致意外行为。

**DB2_MEM_TUNING_RANGE**

- 操作系统: 所有操作系统
- 缺省值: NULL; 值: 一系列百分比 *n*、*m*，其中 *n=minfree*，而 *m=maxfree*
- DB2 实例保留可用的物理内存量十分重要，这是因为此数量指示了同一机器上运行的其他应用程序能够使用的内存量。当启用了数据库共享内存自调整功能时，给定实例保留可用的物理内存量取决于该实例（及其活动数据库）的内存需求。当一个实例紧急需要更多内存时，它将分配内存，直到系统上

的可用物理内存量达到 *minfree* 指定的百分比为止。当该实例不需要那么多内存时，它将维护较大的可用物理内存量，此数量由 *maxfree* 以百分比形式指定。因此，要求对 *minfree* 设置的值必须小于 *maxfree* 值。

如果未设置此变量，那么 DB2 数据库管理器就会根据服务器上的内存量计算 *minfree* 和 *maxfree* 的值。除非正在运行自调整内存管理器（STMM）并将 **database_memory** 设置为 AUTOMATIC，否则此变量的设置不起作用。

**DB2_MMAP_READ**

- 操作系统：AIX
- 缺省值：OFF；值：ON 或 OFF
- 此变量与 **DB2_MMAP_WRITE** 一起使用，以允许 DB2 数据库系统使用 mmap 作为 I/O 的一个替代方法。

  将这些变量设置为 ON 时，对 DB2 缓冲池读写的数据将绕过 AIX 内存高速缓存。如果 DB2 缓冲池相对较小，并且您不能或选择不增大此缓冲池的大小，那么应该通过将 **DB2_MMAP_READ** 和 **DB2_MMAP_WRITE** 设置为 OFF 来利用 AIX 内存高速缓存。

**DB2_MMAP_WRITE**

- 操作系统：AIX
- 缺省值：OFF；值：ON 或 OFF
- 此变量与 **DB2_MMAP_READ** 一起使用，以允许 DB2 数据库系统使用 mmap 作为 I/O 的一个替代方法。

  将这些变量设置为 ON 时，对 DB2 缓冲池读写的数据将绕过 AIX 内存高速缓存。如果 DB2 缓冲池相对较小，并且您不能或选择不增大此缓冲池的大小，那么应该通过将 **DB2_MMAP_READ** 和 **DB2_MMAP_WRITE** 设置为 OFF 来利用 AIX 内存高速缓存。

**DB2_NO_FORK_CHECK**

- 操作系统：UNIX
- 缺省值：OFF；值：ON 或 OFF
- 当启用此注册表变量时，DB2 运行时客户机将使确定当前过程是否是派生调用的结果而进行的检查次数最少。这样可提高不使用 fork() API 的 DB2 应用程序的性能。

**注**：建议您不要使用此变量，将来的发行版中会将其除去。由于在启动或新派生进程时会高速缓存当前进程标识 (pid)，所以此变量不是必需的。

**DB2NTMEMSIZE**

- 操作系统：Windows
- 缺省值：（随内存段变化）
- Windows 要求在 DLL 初始化时保留所有共享内存段，以保证在不同的进程之间的地址匹配。**DB2NTMEMSIZE** 允许用户在必要时覆盖 Windows 上的 DB2 缺省值。在大多数情况下，缺省值应足够使用。内存段、缺省大小和替换选项为:

  1. 并行 FCM 缓冲区：缺省大小为 512 MB（在 32 位平台上）或 4.5 GB（在 64 位平台上）；替换选项为 FCM:<字节数>

2. 受防护方式的通信：缺省大小为 80 MB（在 32 位平台上）或 512 MB（在 64 位平台上）；替换选项为 APLD:<字节数>

通过用分号（;）来隔开替换选项，可替换多个段。例如，在 32 位版本的 DB2 上，要将 FCM 缓冲区设置为 1 GB 并将受防护的存储过程限制为 256 MB，请使用：

```
db2set DB2NTMEMSIZE=FCM:1073741824;APLD:268435456
```

**DB2NTNOCACHE**

- 操作系统：Windows
- 缺省值：OFF；值：ON 或 OFF
- **DB2NTNOCACHE** 注册表变量指定 DB2 数据库系统是否使用 NOCACHE 选项打开数据库文件。如果 **DB2NTNOCACHE**=ON，那么不会执行文件系统高速缓存。如果 **DB2NTNOCACHE**=OFF，那么操作系统将高速缓存 DB2 文件。这适用于除包含长字段或 LOB 的文件以外的所有数据。消除系统高速缓存使数据库有更多内存可用，这样可以增加缓冲池或排序堆。

  在 Windows 中，在打开文件时对文件进行高速缓存，这是缺省行为。为文件中的每 1 GB 从系统池保留 1 MB。使用此注册表变量来覆盖高速缓存的无正式文件的 192 MB 限制。当达到该高速缓存限制时，给出资源不够错误。

**注**：从版本 8.2 起就不推荐使用 **DB2NTNOCACHE**，将来的发行版中会将其除去。通过使用 CREATE TABLESPACE 和 ALTER TABLESPACE SQL 语句，可以对表空间容器获得相同好处。

**DB2NTPRICLASS**

- 操作系统：Windows
- 缺省值：NULL；值：R、H 或任何其他值
- 设置 DB2 实例（程序 DB2SYSCS.EXE）的优先级类别。有三种优先级类别：
  - NORMAL_PRIORITY_CLASS（缺省值优先级类别）
  - REALTIME_PRIORITY_CLASS（使用 R 设置）
  - HIGH_PRIORITY_CLASS（使用 H 设置）

  此变量与使用 **DB2PRIORITIES** 设置的个别线程优先级一起使用，以确定此系统中 DB2 线程相对于其他线程的绝对优先级。

  **注**：不推荐使用 **DB2NTPRICLASS**，该参数只应在服务建议中使用。使用 DB2 服务类来调整代理程序优先级和预取优先级。使用此变量时应当小心。误用可能会对整个系统的性能有负面影响。

  有关更多信息，请参阅 Win32 文档中的 SetPriorityClass() API。

**DB2NTWORKSET**

- 操作系统：Windows
- 缺省值：1,1
- 用于修改 DB2 数据库管理器可用的最小和最大工作集大小。缺省值情况下，当 Windows 未处于页面调度情况时，进程的工作集可按需要增大。但是，当发生页面调度时，进程可拥有的最大工作集大约是 1 MB。**DB2NTWORKSET** 允许您重设此缺省行为。

使用语法 **DB2NTWORKSET**=*min*, *max* 指定 **DB2NTWORKSET**，其中 *min* 和 *max* 以兆字节表达。

**DB2_OVERRIDE_BPF**

- 操作系统：所有操作系统
- 缺省值：未设置；值：正数页数或 <entry>[;<entry>...]，其中 <entry>=<buffer pool ID>,<number of pages>
- 此变量以页为单位指定在激活数据库、前滚恢复或崩溃恢复时要创建的缓冲池大小。如果内存约束导致在激活数据库、前滚恢复或崩溃恢复时出现故障，那么此选项很有用。出现内存不足可能是因为实内存短缺（这种情况很少发生）；或者因为数据库管理器试图在没有精确配置的缓冲池的情况下分配大的缓冲池。例如，如果数据库管理器连 16 页的最小缓冲池都无法建立，那么尝试使用此环境变量来指定一个更小的页数。为此变量提供的值将覆盖当前的缓冲池大小。

  还可以使用 <entry>[;<entry>...]（其中 <entry>=<buffer pool ID>,<number of pages>）来临时更改所有缓冲池或部分缓冲池的大小以使它们可以启动。

**DB2_PINNED_BP**

- 操作系统：AIX、HP-UX 和 Linux
- 缺省值：NO；值：YES 或 NO
- 将此变量设置为 YES 会导致 DB2 请求操作系统固定 DB2 的数据库共享内存。在配置 DB2 以固定数据库共享内存时，应注意确保不要过量使用系统，否则操作系统的内存管理灵活性会降低。

  在 Linux 上，除了要求修改此注册表变量以外，还需要 `libcap.so.1` 库。

  如果将此变量设置为 YES，那么将无法启用数据库共享内存自调整功能（通过将 **database_memory** 配置参数设置为 AUTOMATIC 来激活此功能）。

  对于 AIX 操作系统，要将数据库内存保留与中等页面大小支持配合使用（它是缺省行为），请通过以 root 用户权限登录和发出下列命令来确保实例拥有者具有 CAP_BYPASS_RAC_VMM 和 CAP_PROPAGATE 功能:

```
chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE <instance_owner_user_id>
```

  对于 64 位环境中的 HP-UX，除了修改此注册表变量外，还必须给予 DB2 实例组 MLOCK 特权。为此，具有 root 用户访问权限的用户要执行下列操作:

  1. 将 DB2 实例添加至 `/etc/privgroup` 文件中。例如，如果 DB2 实例组属于 `db2iadm1` 组，那么必须将下面这一行添加到 `/etc/privgroup` 文件中:

     ```
     db2iadm1 MLOCK
     ```

  2. 发出以下命令:

     ```
     setprivgrp -f /etc/privgroup
     ```

**DB2PRIORITIES**

- 操作系统：所有操作系统
- 值的设置是与平台相关
- 控制 DB2 进程和线程的优先级。

**注**: 不推荐使用 **DB2PRIORITIES**，该参数只应在服务建议中使用。使用 DB2 服务类来调整代理程序优先级和预取优先级。

### DB2_RCT_FEATURES

- 操作系统: 所有操作系统
- 缺省值: NULL。值: GROUPUPDATE=[ON|OFF]。GROUPUPDATE 的缺省值为 OFF。
- 当仅指定了前导列和键序列列子集上的等价谓词时，此变量允许对搜索式 UPDATE 语句执行经过优化和缩减的更新处理，而此语句将范围集群表中的多个行作为目标。由于在一个页面上只是更新所有行的单个日志记录，而不是更新每一行的日志记录，因此日志记录也会减少。

  用法:

  ```
  db2set DB2_RCT_FEATURES=GROUPUPDATE=ON
  ```

  注意: 此注册表变量在版本 9.5 修订包 3 中可用。一旦启用了此注册表变量，就不能逆向迁移到较低版本的修订包。此外，当 UPDATE 语句的目标表使用 DATA CAPTURE CHANGES、触发器、可变长度列，具有任何辅助索引或者组更新将修改任何主键列或外键列时，将不会执行组更新。

### DB2_RESOURCE_POLICY

- 操作系统: AIX 5 或更高版本，除了 zSeries（32 位）之外的所有 Linux 和 Windows Server 2003 或更高版本
- 缺省值: 未设置; 值: 指向配置文件的有效路径
- 定义如下资源策略: 可以使用该策略来限制 DB2 数据库使用的操作系统资源，或者该策略包含用于将特定操作系统资源指定给特定 DB2 数据库对象的规则。例如，在 AIX、Linux 或 Windows 操作系统上，可使用此注册表变量来限制 DB2 数据库系统使用的一组处理器。资源控制的范围根据操作系统的不同而变化。

  在启用了 AIX NUMA 和 Linux NUMA 的机器上，可以定义指定 DB2 数据库系统使用的资源集的策略。使用资源集绑定时，将每个单独的 DB2 进程绑定至特定资源集。这对于一些性能调整方案很有用。

  可以设置该注册表变量来指示指向这样一个配置文件的路径: 该配置文件定义用于将 DB2 进程绑定至操作系统资源的策略。资源策略使您可以指定一组操作系统资源来限制 DB2 数据库系统。每个 DB2 进程都绑定至该资源集中的单个资源。资源分配以循环方式进行。

  样本配置文件:

  示例 1: 将所有 DB2 进程绑定至 CPU 1 或 3。

  ```
  <RESOURCE_POLICY>
   <GLOBAL_RESOURCE_POLICY>
     <METHOD>CPU</METHOD>
    <RESOURCE_BINDING>
    <RESOURCE>1</RESOURCE>
    </RESOURCE_BINDING>
    <RESOURCE_BINDING>
  ```

```
    <RESOURCE>3</RESOURCE>
  </RESOURCE_BINDING>
 </GLOBAL_RESOURCE_POLICY>
</RESOURCE_POLICY>
```

示例 2: （仅适用于 AIX）将 DB2 进程绑定至下列其中一个资源集: sys/node.03.00000、sys/node.03.00001、sys/node.03.00002 和 sys/node.03.00003

```
<RESOURCE_POLICY>
 <GLOBAL_RESOURCE_POLICY>
   <METHOD>RSET</METHOD>
  <RESOURCE_BINDING>
    <RESOURCE>sys/node.03.00000</RESOURCE>
  </RESOURCE_BINDING>
  <RESOURCE_BINDING>
    <RESOURCE>sys/node.03.00001</RESOURCE>
  </RESOURCE_BINDING>
  <RESOURCE_BINDING>
    <RESOURCE>sys/node.03.00002</RESOURCE>
  </RESOURCE_BINDING>
  <RESOURCE_BINDING>
    <RESOURCE>sys/node.03.00003</RESOURCE>
  </RESOURCE_BINDING>
 </GLOBAL_RESOURCE_POLICY>
</RESOURCE_POLICY>
```

**注**: （仅适用于 AIX）使用 RSET 方法需要具有 CAP_NUMA_ATTACH 能力。

示例 3: （仅适用于 Linux）将与 SAMPLE 数据库关联的缓冲池标识 2 和 3 中的所有内存绑定至 NUMA 节点 3。此外，将数据库总内存的 80% 用于绑定至 NUMA 节点 3，并留出 20% 分布在所有节点中作为非特定于缓冲池的内存。

```
<RESOURCE_POLICY>
 <DATABASE_RESOURCE_POLICY>
   <DBNAME>sample</DBNAME>
   <METHOD>NODEMASK</METHOD>
  <RESOURCE_BINDING>
    <RESOURCE>3</RESOURCE>
    <DBMEM_PERCENTAGE>80</DBMEM_PERCENTAGE>
   <BUFFERPOOL_BINDING>
     <BUFFERPOOL_ID>2</BUFFERPOOL_ID>
     <BUFFERPOOL_ID>3</BUFFERPOOL_ID>
   </BUFFERPOOL_BINDING>
  </RESOURCE_BINDING>
 </DATABASE_RESOURCE_POLICY>
</RESOURCE_POLICY>
```

示例 4: （仅适用于 Linux 和 Windows）定义由 CPU 掩码 0x0F 和 0xF0 指定的两个截然不同的处理器集合。将 DB2 进程和缓冲池标识 2 绑定至处理器集合 0x0F，而将 DB2 进程和缓冲池标识 3 绑定至处理器集合 0xF0。对于每个处理器集合，将数据库内存总量的 50% 用于绑定。

当期望建立处理器与 NUMA 节点之间的映射时，此资源策略很有用。下面是这种情况的一个示例: 一个系统中有 8 个处理器和两个 NUMA 节点，其中处理器 0 到 3 属于 NUMA 节点 0，而处理器 4 到 7 属于 NUMA 节点 1。此资源策略允许在进行处理器绑定的同时隐式维护内存的局部性（即，CPU 方法和 NODEMASK 方法混合使用）。

```
<RESOURCE_POLICY>
 <DATABASE_RESOURCE_POLICY>
   <DBNAME>sample</DBNAME>
<METHOD>CPUMASK</METHOD>
  <RESOURCE_BINDING>
<RESOURCE>0x0F</RESOURCE>
<DBMEM_PERCENTAGE>50</DBMEM_PERCENTAGE>
   <BUFFERPOOL_BINDING>
     <BUFFERPOOL_ID>2</BUFFERPOOL_ID>
   </BUFFERPOOL_BINDING>
  </RESOURCE_BINDING>
  <RESOURCE_BINDING>
   <RESOURCE>0xF0</RESOURCE>
<DBMEM_PERCENTAGE>50</DBMEM_PERCENTAGE>
   <BUFFERPOOL_BINDING>
     <BUFFERPOOL_ID>3</BUFFERPOOL_ID>
   </BUFFERPOOL_BINDING>
  </RESOURCE_BINDING>
 </DATABASE_RESOURCE_POLICY>
</RESOURCE_POLICY>
```

**注**: 使用 RSET 方法需要具有 CAP_NUMA_ATTACH 能力，并且在 Linux 上不受支持。

**DB2_RESOURCE_POLICY** 注册表变量指定的配置文件接受 SCHEDULING_POLICY 元素。在某些平台上，可以使用 SCHEDULING_POLICY 元素来选择

– DB2 服务器使用的操作系统调度策略

  可以使用 **DB2NTPRICLASS** 注册表变量来为 AIX 上的 DB2 和 Windows 上的 DB2 设置操作系统调度策略。

– 各个 DB2 服务器代理程序使用的操作系统优先级

此外，可以使用注册表变量 **DB2PRIORITIES** 和 **DB2NTPRICLASS** 来控制操作系统调度策略和设置 DB2 代理程序优先级。但是，如果在资源策略配置文件中指定 SCHEDULING_POLICY 元素，就可以在单一位置指定调度策略和相关代理程序优先级。

示例 1: 选择 AIX SCHED_FIFO 调度策略，提高 DB2 日志写程序和阅读器进程的优先级。

```
<RESOURCE_POLICY>
 <SCHEDULING_POLICY>
  <POLICY_TYPE>SCHED_FIFO</POLICY_TYPE>
   <PRIORITY_VALUE>60</PRIORITY_VALUE>

  <EDU_PRIORITY>
    <EDU_NAME>db2loggr</EDU_NAME>
    <PRIORITY_VALUE>56</PRIORITY_VALUE>
  </EDU_PRIORITY>

  <EDU_PRIORITY>
    <EDU_NAME>db2loggw</EDU_NAME>
<PRIORITY_VALUE>56</PRIORITY_VALUE>
  </EDU_PRIORITY>
 </SCHEDULING_POLICY>
</RESOURCE_POLICY>
```

示例 2: 在 Windows 上替换 DB2NTPRICLASS=H。

```
<RESOURCE_POLICY>
 <SCHEDULING_POLICY>
   <POLICY_TYPE>HIGH_PRIORITY_CLASS</POLICY_TYPE>
 </SCHEDULING_POLICY>
</RESOURCE_POLICY>
```

### DB2_SET_MAX_CONTAINER_ SIZE

- 操作系统: 所有操作系统
- 缺省值: 未设置; 值: -1 或大于 65536 个字节的任何正整数
- 此注册表变量允许您限制启用了 AutoResize 功能的自动存储器表空间的各个容器大小。

  **注**: 尽管您可以按字节、千字节或兆字节来指定 **DB2_SET_MAX_CONTAINER_SIZE**, 但是 db2set 会按字节数来指示它的值。
- 如果该值设置为 -1, 那么对容器大小没有限制。

### DB2_SKIPDELETED

- 操作系统: 所有操作系统
- 缺省值: OFF; 值: ON 或 OFF
- 当启用了此注册表变量时, 它允许使用“游标稳定性”或“读稳定性”隔离级别的语句（在索引访问期间）无条件地跳过已删除的键和（在表访问期间）跳过已删除的行。当启用了 **DB2_EVALUNCOMMITTED** 时, 将自动跳过已删除的行, 但是, 除非还启用了 **DB2_SKIPDELETED**, 否则将不会跳过 2 类索引中未落实的伪删除键。

  此注册表变量不会影响 DB2 目录表上的游标的行为。

  此注册表变量用 db2start 命令激活。

### DB2_SKIPINSERTED

- 操作系统: 所有操作系统
- 缺省值: OFF; 值: ON 或 OFF
- 当启用了 **DB2_SKIPINSERTED** 注册表变量时, 它允许使用“游标稳定性”或“读稳定性”隔离级别的语句跳过未落实的已插入行, 就好像从未插入这些行一样。此注册表变量不会影响 DB2 目录表上的游标的行为。此注册表变量是在数据库启动时激活的, 而跳过未落实的已插入行的决定是在语句编译时或构建时做出的。

  **注**: 跳过已插入行的行为与具有暂挂转出清除的表不兼容。因此, 扫描程序可能等待 RID 上的锁定, 这样做只是为了发现该 RID 是否是已转出块的一部分。

### DB2_SMS_TRUNC_TMPTABLE_ THRESH

- 操作系统: 所有操作系统
- 缺省值: 0; 值: -1 或 0-*n*, 其中 *n* = SMS 表空间容器中针对每个临时表要维护的扩展数据块的数目
- 此变量指定表示将在 SMS 表空间中维护的临时表的文件的最小文件大小阈值。

缺省情况下，此变量设置为 0，它表示不会执行特殊阈值处理。但是，一旦不再需要临时表，该文件就会被截断为 0 个扩展数据块。

当此变量的值大于 0 时，将维护更大的文件。这样可减少每次使用临时表时删除并重新创建文件所造成的一部分系统开销。

如果此变量设置为 -1，那么不会截断该文件，并且允许它无限增长，只不过会受到系统资源的限制。

**DB2_SORT_AFTER_TQ**

- 操作系统：所有操作系统
- 缺省值：NO；值：YES 或 NO
- 指定当接收端要求对数据排序并且接收节点数与发送节点数相等时，优化器如何在分区数据库环境中使用定向表队列。

  当 **DB2_SORT_AFTER_TQ**=NO 时，优化器往往会在发送端排序，而在接收端合并行。

  当 **DB2_SORT_AFTER_TQ**=YES，优化器往往会发送未排序的行，在接收端不合并，而在接收完所有的行之后才在接收端对这些行进行排序。

**DB2_SELUDI_COMM_BUFFER**

- 操作系统：所有操作系统
- 缺省值：OFF；值：ON 或 OFF
- 此变量在通过 UPDATE、INSERT 或 DELETE（UDI）查询中的 SELECT 来处理分块游标时使用。当启用此注册表变量时，它会阻止将查询结果存储在临时表中。但是，在对 UDI 查询的 SELECT 的分块游标执行 OPEN 处理期间，DB2 数据库系统会尝试将查询的整个结果直接缓存到通信缓冲区内存区域中。

  **注**：如果通信缓冲区空间不足以容纳查询的整个结果，那么发出 SQLCODE -906 错误并回滚事务。有关分别调整本地和远程应用程序的通信缓冲区内存区域大小的信息，请参阅 **aslheapsz** 和 **rqrioblk** 数据库管理器配置参数。

  在分区数据库环境中或者启用了分区内并行性时，不支持此注册表变量。

**DB2_TRUSTED_BINDIN**

- 操作系统：所有操作系统
- 缺省值：OFF；值：OFF、ON 或 CHECK
- 当启用了 **DB2_TRUSTED_BINDIN** 时，会加速执行不受防护的嵌入式存储过程中包含主变量的查询语句。

  当启用了此变量时，在绑定不受防护的嵌入式存储过程中包含的 SQL 和 XQuery 语句期间，不会将外部 SQLDA 格式转换为内部 DB2 格式。这将加速处理嵌入式 SQL 和 XQuery 语句。

  当启用了此变量时，不受防护的嵌入式存储过程中不支持下列数据类型:

  – SQL_TYP_DATE
  – SQL_TYP_TIME
  – SQL_TYP_STAMP

- SQL_TYP_CGSTR
- SQL_TYP_BLOB
- SQL_TYP_CLOB
- SQL_TYP_DBCLOB
- SQL_TYP_CSTR
- SQL_TYP_LSTR
- SQL_TYP_BLOB_LOCATOR
- SQL_TYP_CLOB_LOCATOR
- SQL_TYP_DCLOB_LOCATOR
- SQL_TYP_BLOB_FILE
- SQL_TYP_CLOB_FILE
- SQL_TYP_DCLOB_FILE
- SQL_TYP_BLOB_FILE_OBSOLETE
- SQL_TYP_CLOB_FILE_OBSOLETE
- SQL_TYP_DCLOB_FILE_OBSOLETE

如果遇到了这些数据类型，那么会返回 SQLCODE -804 和 SQLSTATE 07002。

**注**: 输入主变量的数据类型和长度必须与相应元素的内部数据类型和长度精确匹配。对于主变量，将始终满足此要求。但是，对于参数标记，必须注意确保使用相匹配的数据类型。可以使用 CHECK 选项来确保所有输入主变量的数据类型和长度都匹配，但是此选项将不利于性能提高。

**注**: 建议您不要使用 **DB2_TRUSTED_BINDIN**，在以后的发行版中会将其除去。

**DB2_USE_ALTERNATE_PAGE_CLEANING**

- 操作系统: 所有操作系统
- 缺省值: 未设置; 值: ON 或 OFF
- 此变量指定 DB2 数据库使用页清除算法的另一种方法还是使用页清除的缺省方法。当此变量设置为 ON 时，DB2 系统会将更改的页写入磁盘，从而保持在 LSN_GAP 前面并且主动查找牺牲页。这样做允许页清除程序更好地利用可用的磁盘 I/O 带宽。当此变量设置为 ON 时，由于 **chngpgs_thresh** 数据库配置参数不控制页清除程序的活动，所以它不再相关。

**DB2_USE_FAST_PREALLOCATION**

- 操作系统: VeritasVxFS 或 JFS2 文件系统上的 AIX、Linux 和 Solaris
- 缺省值: ON; 值: ON 或 OFF
- 允许 Veritas 快速分配文件系统功能部件或 AIX JFS2 保留表空间，并加速完成创建或改变大型表空间和数据库恢复操作这一过程。实现此速度改善所增加的成本较低，也就是增加了在运行时期间插入行时分配实际空间所产生的成本。

  要禁用快速预分配，请将 **DB2_USE_FAST_PREALLOCATION** 设置为 OFF。这可能会提高运行时性能; 但是，在某些操作系统上（尤其是 AIX），

当在同一表空间中执行大量插入和选择操作时，创建表空间的速度就会降低，并且数据库恢复时间将延长。注意，一旦禁用了快速预分配，就必须恢复数据库。

在 DB2 版本 9.5 修订包 6 中提供了此变量。

**DB2_USE_IOCP**

- 操作系统：AIX 5.3 TL9 SP2 或 AIX 6.1 TL2
- 缺省值：OFF；值：OFF 或 ON
- 请在启用此注册表变量之前配置 IOCP。
- 通过使用此变量，就可以在提交和收集异步 I/O (AIO) 请求时使用 AIX I/O 完成端口 (IOCP)。此功能用来提高非一致性内存访问 (NUMA) 环境中的性能，这是因为它可以避免进行远程内存访问。版本 9.5 修订包 3 中提供了此性能变量。

## 其他变量

**DB2ADMINSERVER**

- 操作系统：Windows 和 UNIX
- 缺省值：NULL
- 指定 DB2 管理服务器。

**DB2_ATS_ENABLE**

- 操作系统：所有操作系统
- 缺省值：NULL；值：YES/TRUE/ON/1 或者 NO/FALSE/OFF/0
- 此变量用于控制管理任务调度程序是否正在运行。缺省情况下，管理任务调度程序处于禁用状态。当此调度程序处于禁用状态时，可以使用内置过程和视图来定义和修改任务，但是此调度程序将不执行任务。

**DB2AUTH**

- 操作系统：所有操作系统
- 值：TRUSTEDCLIENT_SRVRENC、TRUSTEDCLIENT_DATAENC、DISABLE_CHGPASS 和 OSAUTHDB
- 此变量允许您调整用户认证的行为。
  - TRUSTEDCLIENT_SRVRENC：此值强制不可信客户机使用 SERVER_ENCRYPT。
  - TRUSTEDCLIENT_DATAENC：此值强制不可信客户机使用 DATA_ENCRYPT。
  - DISABLE_CHGPASS：此值禁用从客户机更改密码的能力。
  - OSAUTHDB：从 DB2 版本 9.5 修订包 4 开始，此值指示 DB2 在 AIX 操作系统上使用用户的认证和组设置。从修订包 5 开始，已经将透明 LDAP 支持扩展到 Linux、HP-UX 和 Solaris 操作系统。LDAP 服务器可以是下列任何一项：
    - IBM Tivoli Directory Server（ITDS）
    - Microsoft Active Directory（MSAD）
    - Sun One Directory Server

**DB2CLIINIPATH**

- 操作系统: 所有操作系统
- 缺省值: NULL
- 用于替换 DB2 CLI/ODBC 配置文件（db2cli.ini）的缺省路径并指定客户机上另一个位置。指定的值必须是客户机系统上的有效路径。

**DB2_COMMIT_ON_EXIT**

- 操作系统: UNIX
- 缺省值: OFF; 值: OFF/NO/0 或 ON/YES/1
- 在 UNIX 操作系统上，在 DB2 UDB 版本 8 之前的版本中，当成功退出应用程序时，DB2 将落实任何余下的未完成的事务。在 DB2 UDB 版本 8 中，该行为更改为退出时将回滚未完成的事务。此注册表变量允许具有依赖于先前行为的嵌入式 SQL 应用程序的用户在版本 9 中继续启用该程序。此注册表变量不影响 JDBC、CLI 和 ODBC 应用程序。

  请注意，不推荐使用此注册表变量，并且将来的发行版中不再支持“退出时落实”这种行为。用户应该确定他们在版本 9 之前开发的任何应用程序是否继续依赖于此功能，并根据需要将适当的显式 COMMIT 或 ROLLBACK 语句添加至应用程序。如果打开该注册表变量，那么应该注意不要实施在退出之前未能显式 COMMIT（落实）的新应用程序。

  大多数用户都应该将此注册表变量保留为缺省设置。

**DB2_CREATE_DB_ON_PATH**

- 操作系统: Windows
- 缺省值: NULL; 值: YES 或 NO
- 将此注册表变量设置为 YES，以支持将一个路径（以及驱动器）用作数据库路径。创建数据库时、设置数据库管理器配置参数 **dftdbpath** 时以及恢复数据库时，将检查 **DB2_CREATE_DB_ON_PATH** 的设置。标准数据库路径长度可达 215 个字符。

  创建或恢复数据库时，如果没有设置 **DB2_CREATE_DB_ON_PATH**（或将其设置为 NO）且指定了数据库路径，那么会返回错误 SQL1052N。

  如果没有设置 **DB2_CREATE_DB_ON_PATH**（或将其设置为 NO）且更新了 **dftdbpath** 数据库管理器配置参数，那么会返回错误 SQL5136N。

  **注意:**
  **如果使用路径支持来创建新数据库，那么使用 db2DbDirGetNextEntry() API 编写的版本 9.1 之前或更低版本的应用程序可能无法正常使用。请参阅 http://www.ibm.com/software/data/db2/support/db2_9/，以了解各种方案和正确操作过程的详细信息。**

**DB2DEFPREP**

- 操作系统: 所有操作系统
- 缺省值: NO; 值: ALL、YES 或 NO

- 对在 DEFERRED_PREPARE 预编译选项可用之前已预编译的应用程序，模拟此选项的运行时行为。例如，如果 DB2 V2.1.1 或更低版本的应用程序在 DB2 V2.1.2 或更高版本的环境中运行，那么可使用 **DB2DEFPREP** 来指示期望的“延迟准备”行为。

**注**: 不推荐使用 **DB2DEFPREP**，将来的发行版中会将其除去。只有使用旧版本 DB2（DEFERRED_PREPARE 预编译选项在其中不可用）的用户才需要此变量。

**DB2_DISABLE_FLUSH_LOG**

- 操作系统: 所有操作系统
- 缺省值: OFF; 值: ON 或 OFF
- 指定在联机备份完成时，是否禁止关闭活动日志文件。

  当联机备份完成时，最后一个活动日志文件将被截断、关闭并且可以归档。这样就可确保您的联机备份有一组完整的归档日志可用于恢复。如果您担心这样会浪费部分日志序号（LSN）空间，那么可以禁用关闭最后一个活动日志文件。每次截断活动日志文件时，LSN 按截断空间量的比例递增。如果您每天都要执行大量联机备份，那么可以禁止关闭最后一个活动日志文件。

  如果您在完成联机备份后不久就收到日志已满的消息，那么可能也需要禁止关闭最后一个活动日志文件。当一个日志文件被截断时，剩余的活动日志空间根据被截断的日志大小的比例递增。回收截断的日志文件时，将释放活动日志空间。日志文件处于不活动状态后不久，就会进行回收。在这两个事件之间的短暂时间间隔内，您将收到日志已满的消息。

  在任何备份（包括日志）期间，此注册表变量将被忽略，因为活动日志文件必须被截断并关闭才能使备份包括日志。

**DB2CONNECT_DISCONNECT_ON_INTERRUPT**

- 操作系统: 所有操作系统
- 缺省值: NO; 值: YES/TRUE/1 或 NO/FALSE/0
- 当此变量设置为 YES（TRUE 或 1）时，它指定在发生中断时，应立即断开与版本 8（或更高版本）的 DB2 通用数据库 z/OS 服务器的连接。可以在下列配置中使用此变量:
  - 如果正在将 DB2 客户机与版本 8（或更高版本）DB2 UDB z/OS 服务器配合使用，那么在该客户机上将 **DB2CONNECT_DISCONNECT_ON_INTERRUPT** 设置为 YES。
  - 如果正在通过 DB2 Connect 网关对版本 8（或更高版本）DB2 UDB z/OS 服务器运行 DB2 客户机，那么在该网关上将 **DB2CONNECT_DISCONNECT_ON_INTERRUPT** 设置为 YES。

**DB2_DISPATCHER_PEEKTIMEOUT**

- 操作系统: 所有操作系统
- 缺省值: 1; 值: 0 到 32767 秒; 0 表示立即超时
- **DB2_DISPATCHER_PEEKTIMEOUT** 允许您调整分派器等待客户机的连接请求直至将客户机挂起至代理程序的时间，该时间以秒计。在大多数情况下，您不需要调整此注册表变量。此注册表变量只影响启用了 DB2 Connect 连接集中器的实例。

此注册表变量和 **DB2_SERVER_CONTIMEOUT** 注册表变量都配置在连接期间处理新客户机的方式。如果有很多客户机以很慢的速度与实例进行连接，那么分派器将对每个客户机分配长达 1 秒的超时时间；如果很多客户机同时进行连接，那么会导致分派器瓶颈。如果具有多个活动数据库的实例连接速度很慢，那么 **DB2_DISPATCHER_PEEKTIMEOUT** 可能小于 0。降低 **DB2_DISPATCHER_PEEKTIMEOUT** 会导致分派器只观察已有客户机的连接请求，而不等待该连接请求到达。如果设置了无效值，那么将使用缺省值。此注册表变量不是动态的。

**DB2_DJ_INI**

- 操作系统：所有操作系统
- 缺省值：
  - UNIX：*db2_instance_directory*/cfg/db2dj.ini
  - Windows：*db2_install_directory*\cfg\db2dj.ini
- 指定联合配置文件的绝对路径名，例如：db2set DB2_DJ_INI=$HOME/sqllib/cfg/my_db2dj.ini，此文件包含数据源环境变量的设置。这些环境变量由 Informix® 包装器和 WebSphere Federation Server 提供的包装器使用。

  下面是一个样本联合配置文件：

  ```
  INFORMIXDIR=/informix/client_sdk
  INFORMIXSERVER=inf93
  ORACLE_HOME=/usr/oracle9i
  SYBASE=/sybase/V12
  SYBASE_OCS=OCS-12_5
  ```

  下列限制适用于 db2dj.ini 文件：
  - 各个条目必须遵循以下格式：*evname=value*，其中 *evname* 是环境变量的名称，而 *value* 是它的值。
  - 环境变量名称的最大长度为 255 字节。
  - 环境变量值的最大长度为 765 字节。

  除非将数据库管理器参数 **FEDERATED** 设置为 YES，否则将忽略此变量。

**DB2DMNBCKCTLR**

- 操作系统：Windows
- 缺省值：NULL；值：？或域名
- 如果您知道 DB2 服务器充当备份域控制器的域的名称，那么设置 **DB2DMNBCKCTLR**=DOMAIN_NAME。DOMAIN_NAME 必须是大写形式。要让 DB2 确定本地机器是其备份域控制器的域，设置 **DB2DMNBCKCTLR**=?。如果 **DB2DMNBCKCTLR** 概要文件变量未设置或设置为空白，那么 DB2 在主域控制器上执行认证。

  **注：** 缺省情况下，DB2 不使用现有的备份域控制器，因为备份域控制器可以脱离与主域控制器的同步，从而导致安全漏洞。当更新了主域控制器的安全性数据库但未将该更改传播至备份域控制器时，那么可能导致不同步。如果存在网络等待时间或计算机浏览器服务未运行，也可能发生此情况。

**注：** 不推荐使用 **DB2DMNBCKCTLR**，在以后的发行版中会将其除去。由于 Active Directory 中没有其他备份域控制器，所以不再需要此变量。

**DB2_DOCHOST**

- 操作系统: 所有操作系统
- 缺省值: 未设置（但 DB2 仍会尝试从 IBM Web 站点（publib.boulder.ibm.com/infocenter/db2luw/v9r5）访问信息中心）; 值: http://*hostname*, 其中 *hostname* 是有效的主机名或 IP 地址
- 指定安装了 *DB2 信息中心*的主机名。如果在 DB2 安装向导中选择了自动配置选项，那么在安装 *DB2 信息中心*期间可以自动设置此变量。

**DB2_DOCPORT**

- 操作系统: 所有操作系统
- 缺省值: NULL; 值: 任何有效端口号
- 指定 DB2 帮助系统为 DB2 文档服务时使用的端口号。如果在 DB2 安装向导中选择了自动配置选项，那么在安装 *DB2 信息中心*期间可以自动设置此变量。

**DB2_ENABLE_AUTOCONFIG_DEFAULT**

- 操作系统: 所有操作系统
- 缺省值: NULL; 值: YES 或 NO
- 此变量控制创建数据库时是否自动运行配置顾问程序。如果未设置 **DB2_ENABLE_AUTOCONFIG_DEFAULT**（NULL），那么效果等同于将该变量设置为 YES，因此创建数据库时将运行配置顾问程序。设置此变量后，不需要重新启动实例。如果执行 AUTOCONFIGURE 命令或者运行 CREATE DB AUTOCONFIGURE，那么这些命令将覆盖 **DB2_ENABLE_AUTOCONFIG_DEFAULT** 设置。

**DB2_ENABLE_LDAP**

- 操作系统: 所有操作系统
- 缺省值: NO; 值: YES 或 NO
- 指定是否使用“轻量级目录访问协议”（LDAP）。LDAP 是一种目录服务访问方法。

**DB2_EVMON_EVENT_LIST_SIZE**

- 操作系统: 所有操作系统
- 缺省值: 0（没有限制）; 值: 以 KB/Kb/kb、MB/Mb/mb 或 GB/Gb/gb 计; 虽然此变量没有固定的上限，但它受监视器堆中的可用内存量限制。
- 此注册表变量指定排队等待写入特定事件监视器的最大字节数。达到此限制后，尝试发送事件监视器记录的代理程序将等待，直到队列大小降低到小于此阈值为止。

  **注**: 如果无法从监视器堆分配活动记录，那么这些记录将被删除。为了防止出现这种情况，将 **mon_heap_sz** 配置参数设置为 AUTOMATIC。如果将 **mon_heap_sz** 设置为特定值，那么确保 **DB2_EVMON_EVENT_LIST_SIZE** 设置为更小的值。但是，这些操作不能保证活动记录不被删除，因为监视器堆还用于跟踪其他监视元素。

**DB2_EVMON_STMT_FILTER**

- 操作系统: 所有操作系统
- 缺省值: 未设置; 值:

- ALL: 指示将过滤所有语句事件监视器的输出。此选项是独占选项。
- '*nameA nameB nameC*': 字符串中的每个名称表示要对其过滤记录的事件监视器的名称。如果提供了多个名称，那么必须用一个空格分隔每个名称。DB2 将使所有输入名称为大写。可以指定的事件监视器的最大数目为 32。每个监视器的名称最长可以为 18 个字符。
- '*nameA:op1,op2 nameB:op1,op2 nameC:op1*': 字符串中的每个名称表示要对其过滤记录的事件监视器的名称。每个选项（*op1*、*op2* 等等）表示特定 SQL 操作的整数值映射。指定整数值允许用户确定对哪个事件监视器应用哪些规则。

• 可以使用 **DB2_EVMON_STMT_FILTER** 来减少语句事件监视器写入的记录数。设置了此注册表变量时，它仅导致下列 SQL 操作的记录写入指定的事件监视器:

*表 64. 要用于* **DB2_EVMON_STMT_FILTER** *以将事件监视器输出限制于特定 SQL 操作的值*

| SQL 操作 | 整数值映射 |
|---|---|
| SELECT | 15 |
| EXECUTE | 2 |
| EXECUTE_IMMEDIATE | 3 |
| CLOSE | 6 |
| STATIC COMMIT | 8 |
| STATIC ROLLBACK | 9 |
| CALL | 12 |
| PRE_EXEC | 17 |

所有其他操作将不会出现在语句事件监视器的输出中。为定制要将其记录写至事件监视器的操作集合，请使用整数值。

示例 1:

```
db2set DB2_EVMON_STMT_FILTER= 'mon1 monitor3'
```

在本示例中，mon1 和 monitor3 事件监视器将接收受限应用程序请求列表的记录。例如，如果 mon1 语句事件监视器正在监视的应用程序准备动态 SQL 语句、根据该语句打开游标、从该游标中访存 10,000 行，然后发出游标关闭请求，那么只有关闭请求的记录出现在 mon1 事件监视器输出中。

示例 2:

```
db2set DB2_EVMON_STMT_FILTER='evmon1:3,8 evmon2:9,15
```

在本示例中，evmon1 和 evmon2 将接收受限应用程序请求列表的记录。例如，仅当由 evmon1 语句事件监视器监视的应用程序发出创建语句时，立即执行和静态落实操作才会出现在 evmon1 事件监视器输出中。仅当 evmon2 语句事件监视器监视的应用程序执行涉及选择和静态回滚的 SQL 时，这两个操作才会出现在 evmon2 事件监视器输出中。

**注**: 要了解数据库系统监视器常量的定义，请参阅 `sqlmon.h` 头文件。

### DB2_EXTSECURITY

• 操作系统: Windows

- 缺省值：YES；值：YES 或 NO
- 通过锁定 DB2 对象（系统文件、目录和 IPC 对象）来防止对 DB2 进行未经授权的访问。为避免发生潜在的问题，不应该关闭此注册表变量。如果未设置 **DB2_EXTSECURITY**，那么在 DB2 数据库服务器产品上会认为它的值为 YES，在客户机上会认为它的值为 NO。

**DB2_FALLBACK**

- 操作系统：Windows
- 缺省值：OFF；值：ON 或 OFF
- 此变量允许在回退处理期间强制所有数据库连接断开。它与故障转移支持一起用于带有 Microsoft Cluster Server（MSCS）的 Windows 环境中。如果 **DB2_FALLBACK** 未设置或设置为 OFF，且在回退期间数据库连接一直存在，那么不会导致 DB2 资源脱机。这意味着回退处理将失败。

**DB2_FMP_COMM_HEAPSZ**

- 操作系统：Windows 和 UNIX
- 缺省值：20 MB 或足够的空间来运行 10 个受防护的例程（以较大者为准）。在 AIX 上，缺省值是 256 MB
- 此变量指定用于受防护的例程调用（例如，存储过程或用户定义的函数调用）的池的大小（以 4 KB 页计）。每个受防护的例程使用的空间是 **aslheapsz** 配置参数值的两倍。

  如果正在系统上运行大量的受防护的例程，那么可能需要增大此变量的值。如果只是运行非常少量的受防护的例程，那么可以减小这个变量值。

  将此值设置为 0 表示不创建集合，因此无法调用任何受防护的例程。它还意味着将禁用运行状况监视器和数据库自动维护功能（例如，自动备份、收集统计信息和 REORG），原因是此功能依赖于受防护的例程基础结构。

**DB2_GRP_LOOKUP**

- 操作系统：Windows
- 缺省值：NULL；值：LOCAL、DOMAIN、TOKEN、TOKENLOCAL 或 TOKENDOMAIN
- 此变量指定如何使用 Windows 安全性机制来枚举用户所属于的组。

**DB2_HADR_BUF_SIZE**

- 操作系统：所有操作系统
- 缺省值：2***logbufsz**
- 此变量指定备用日志接收缓冲区大小（以日志页为单位）。如果未设置此变量，那么 DB2 将主要 **logbufsz** 配置参数值的两倍大小用作备用接收缓冲区大小。应该在备用实例中设置此变量。主数据库会将其忽略。

  如果 HADR 同步方式（**hadr_syncmode** 数据库配置参数）设置为 ASYNC，那么在对等状态下，低速的备用数据库可能会导致主数据库上的发送操作延迟，并因此而阻塞主数据库上的事务处理。可以在备用数据库上配置大于缺省日志接收缓冲区的缓冲区用来保存更多未处理的日志数据。这可使得在很短的时间内主数据库生成日志数据的速度比备用数据库使用这些数据的速度更快，而且不会阻塞主数据库上的事务处理。

注:

– 较大的日志接收缓冲区有助于承担主数据库上的峰值事务负载，但是如果备用数据库上的平均重放速度小于主数据库上的记录速度，那么该缓冲区仍将装满。

– 从 DB2 版本 9.1 修订包 5 开始，已经将可以指定的最大大小增大到 4 GB。

**DB2_HADR_NO_IP_CHECK**

- 操作系统: 所有操作系统
- 缺省值: OFF; 值: ON |OFF
- 指定是否对 HADR 连接绕过 IP 检查
- 此变量主要用于在网络地址转换（NAT）环境中对 HADR 连接绕过 IP 交叉检查。建议不要在其他环境中使用此变量，因为它会削弱 HADR 配置的健全检查。缺省情况下，会在建立 HADR 连接时检查本地主机参数和远程主机参数的配置是否一致。主机名称会映射到 IP 地址以进行交叉检查。将执行两项检查:

  – **HADR_LOCAL_HOST** parameter on primary = **HADR_REMOTE_HOST** parameter on standby

  – **HADR_REMOTE_HOST** parameter on primary = **HADR_LOCAL_HOST** parameter on standby

  如果检查失败，连接将关闭。

  当此参数打开时，将不执行 IP 检查。

**DB2_HADR_PEER_WAIT_ LIMIT**

- 操作系统: 所有操作系统
- 缺省值: 0（表示没有限制）; 值: 0 到最大无符号 32 位整数（包含这两者）
- 从 DB2 版本 9.5 修订包 1 开始，设置注册表变量 **DB2_HADR_PEER_WAIT_LIMIT** 后，如果登录 HADR 主数据库因为需要将日志复制至备份而被阻止一定秒数，那么此主数据库将退出对等状态。达到此限制时，主数据库将断开与备用数据库的连接。如果对等窗口被禁用，那么主数据库将进入断开连接状态，而日志记录将恢复。如果对等窗口被启用，那么主数据库将进入断开连接的对等状态，而其中的日志记录将继续被阻止。重新连接或对等窗口到期时，主数据库保持断开连接的对等状态。一旦主数据库脱离断开连接的对等状态，日志记录就会恢复。此参数对备用数据库没有影响。尽管如此，还是建议对主数据库和备用数据库使用相同的值。无效值（并非数字或负数）将解释为“0”，表示没有限制。此参数是静态参数。数据库实例需要重新启动以使此参数生效。

**DB2_HADR_SORCVBUF**

- 操作系统: 所有操作系统
- 缺省值: 操作系统 TCP 套接字接收缓冲区大小; 值: 1024 至 4294967295
- 此变量指定 HADR 连接的操作系统（OS）TCP 套接字接收缓冲区大小，这允许用户定制有别于其他连接的 HADR TCP/IP 行为。某些操作系统会自动对用户指定的值进行取整，或者以静默方式限制用户指定的值的大小。用于 HADR 连接的实际缓冲区大小将记录在 `db2diag.log` 文件。请参阅操作系统

网络调整指南以根据网络流量对此参数进行最佳设置。应将此变量与 **DB2_HADR_SOSNDBUF** 结合使用。

**DB2_HADR_SOSNDBUF**

- 操作系统: 所有操作系统
- 缺省值: 操作系统 TCP 套接字发送缓冲区大小; 值: 1024 至 4294967295
- 此变量指定 HADR 连接的操作系统（OS）TCP 套接字发送缓冲区大小，这允许用户定制有别于其他连接的 HADR TCP/IP 行为。某些操作系统会自动对用户指定的值进行取整，或者以静默方式限制用户指定的值的大小。用于 HADR 连接的实际缓冲区大小将记录在 db2diag.log 文件。请参阅操作系统网络调整指南以根据网络流量对此参数进行最佳设置。应将此变量与 **DB2_HADR_SORCVBUF** 结合使用。

**DB2LDAP_BASEDN**

- 操作系统: 所有操作系统
- 缺省值: NULL; 值: 任何有效的基本专有域名。
- 设置此参数后，DB2 的 LDAP 对象将存储在 LDAP 目录中的

```
CN=System
CN=IBM
CN=DB2
```

指定基本专有名称下。

为 Microsoft Active Directory Server 设置此变量时，请确保在这个专有名称下面定义 CN=DB2、CN=IBM 和 CN=System。

**DB2LDAPCACHE**

- 操作系统: 所有操作系统
- 缺省值: YES; 值: YES 或 NO
- 指定要启用 LDAP 高速缓存。此高速缓存用来编目本地机器上的数据库、节点和 DCS 目录。

要确保高速缓存中具有最新的条目，请执行下列操作:

```
REFRESH LDAP IMMEDIATE ALL
```

此命令从数据库目录和节点目录中更新和除去不正确的条目。

**DB2LDAP_CLIENT_PROVIDER**

- 操作系统: Windows
- 缺省值: NULL（如果可用，使用 Microsoft，否则使用 IBM。）值: IBM 或 Microsoft
- 当在 Windows 环境中运行时，DB2 支持使用 Microsoft LDAP 客户机或 IBM LDAP 客户机来访问 LDAP 目录。这个注册表变量用来显式地选择 DB2 要使用的 LDAP 客户机。

**注**: 要显示此注册表变量的当前值，可使用 db2set 命令:

```
db2set DB2LDAP_CLIENT_PROVIDER
```

**DB2LDAPHOST**

- 操作系统: 所有操作系统

- 缺省值：Null；值：*base_domain_name[:port_number]* 或 *base_domain_name*:SSL:636（使用已启用 SSL 的 LDAP 主机时）
- 指定 LDAP 目录所在位置的主机名和可选端口号，其中 *base_domain_name* 是 TCP/IP 主机名，而 *[:port_number]* 是端口号。

**DB2LDAP_KEEP_CONNECTION**

- 操作系统：所有操作系统
- 缺省值：YES；值：YES 或 NO
- 指定 DB2 是否对它的内部 LDAP 连接句柄进行高速缓存。当此变量设置为 NO 时，DB2 将不把它的 LDAP 连接句柄高速缓存到目录服务器中。这可能会降低性能，但是如果需要使与目录服务器建立的同时处于活动状态的 LDAP 客户机连接数最小，那么可能需要将 **DB2LDAP_KEEP_CONNECTION** 设置为 NO。

  要获得最佳性能，缺省情况下，应将此变量设置为 YES。

  **DB2LDAP_KEEP_CONNECTION** 注册表变量只是作为 LDAP 中的全局级概要文件注册表变量来实现的，因此必须通过对 db2set 命令指定 **-gl** 选项来设置它，如下所示：

  ```
  db2set -gl DB2LDAP_KEEP_CONNECTION=NO
  ```

**DB2LDAP_SEARCH_SCOPE**

- 操作系统：所有操作系统
- 缺省值：DOMAIN；值：LOCAL、DOMAIN 或 GLOBAL
- 指定在轻量级目录访问协议（LDAP）中搜索数据库分区信息或域信息的范围。LOCAL 禁止在 LDAP 目录中进行搜索。DOMAIN 仅在当前目录分区的 LDAP 中进行搜索。GLOBAL 在所有目录分区的 LDAP 中进行搜索，直到找到对象为止。

**DB2_LOAD_COPY_NO_OVERRIDE**

- 操作系统：所有操作系统
- 缺省值：NONRECOVERABLE；值：COPY YES 或 NONRECOVERABLE
- 此变量将把任何 LOAD COPY NO 转换为 LOAD COPY YES 或 NONRECOVERABLE，这取决于该变量的值。此变量适用于 HADR 主数据库以及标准（非 HADR）数据库；HADR 备用数据库将忽略此变量。在 HADR 主数据库上，如果未设置此变量，那么 LOAD COPY NO 将转换为 LOAD NONRECOVERABLE。此变量的值指定不可恢复的装入或复制目标，它使用与 COPY YES 子句相同的语法。

**DB2LOADREC**

- 操作系统：所有操作系统
- 缺省值：NULL
- 用于在前滚期间替换装入副本的位置。如果用户更改了装入副本的物理位置，那么必须在发出前滚之前设置 **DB2LOADREC**。

**DB2LOCK_TO_RB**

- 操作系统：所有操作系统
- 缺省值：NULL；值：STATEMENT

- 指定锁定超时是导致回滚整个事务还是仅回滚当前语句。如果将 **DB2LOCK_TO_RB** 设置为 STATEMENT，那么锁定超时只会导致回滚当前语句。任何其他设置都导致事务回滚。

**DB2_MAP_XML_AS_CLOB_FOR_ DLC**

- 操作系统：所有操作系统
- 缺省值：NO；值：YES 或 NO
- 对于不支持将 XML 作为一种数据类型的客户机（或 DRDA 应用程序请求器），**DB2_MAP_XML_AS_CLOB_FOR_DLC** 注册表变量提供覆盖 XML 值的缺省 DESCRIBE 和 FETCH 行为的能力。缺省值是 NO，它指定对于这些客户机，XML 值的 DESCRIBE 将返回 BLOB（2GB），而 XML 值的 FETCH 会导致将 XML（包括指示 UTF-8 编码的 XML 声明）隐式序列化为 BLOB。

  当值是 YES 时，XML 值的 DESCRIBE 将返回 CLOB（2GB），而 XML 值的 FETCH 会导致将 XML（不包括 XML 声明）隐式序列化为 CLOB。

**注**：不推荐使用 **DB2_MAP_XML_AS_CLOB_FOR_DLC**，将来的发行版中会将其除去。由于大多数访问 XML 值的现有 DB2 应用程序以这种方式对待支持 XML 的客户机，所以不再需要此变量。

**DB2_MAX_LOB_BLOCK_SIZE**

- 操作系统：所有操作系统
- 缺省值：0（没有限制）；值：0 至 21487483647
- 设置在一个块中返回的最大 LOB 或 XML 数据量。这不是硬最大值；如果数据检索期间在服务器上达到此最大值，服务器将完成写当前行的操作，然后再对客户机返回命令（例如 FETCH）应答。

**DB2_MEMORY_PROTECT**

- 操作系统：具有存储密钥支持的 AIX
- 缺省值：NO；值：NO 或 YES
- 此注册表变量启用内存保护功能，该功能使用存储密钥来防止缓冲池中的数据由于无效内存访问而毁坏。内存保护通过确定 DB2 引擎线程应访问缓冲池内存的时间来工作。当 **DB2_MEMORY_PROTECT** 设置为 YES 时，只要 DB2 引擎线程尝试非法访问缓冲池内存，该引擎线程就会被捕获。

  需要启用 **DB2_MEMORY_PROTECT**，陷阱弹性功能 **DB2_THREAD_SUSPENSION** 才能使用。

  **注**：如果 **DB2_LGPAGE_BP** 设置为 YES，您将无法使用内存保护。即使 **DB2_MEMORY_PROTECT** 设置为 YES，DB2 也无法保护缓冲池内存并禁用此功能部件。

**DB2NOEXITLIST**

- 操作系统：所有操作系统
- 缺省值：OFF；值：ON 或 OFF
- 此变量指示无论 **DB2_COMMIT_ON_EXIT** 注册表变量的设置为多少，DB2 都不应该装入出口列表处理程序，并且它不应在应用程序退出时执行落实。

当 **DB2NOEXITLIST** 关闭并且 **DB2_COMMIT_ON_EXIT** 打开时，将自动落实嵌入式 SQL 应用程序的任何未完成事务。建议在应用程序退出时显式添加 COMMIT 或 ROLLBACK 语句。

调用 DB2 出口处理程序时，动态装入和卸装 DB2 库的应用程序可能会在终止前崩溃。这种崩溃可能是因为应用程序尝试调用内存中不存在的函数而产生的。要避免这种情况，请设置 **DB2NOEXITLIST** 注册表变量。

**DB2_NUM_CKPW_DAEMONS**

- 操作系统: UNIX
- 缺省值: 3; 值: 1[:FORK] 至 100[:FORK]
- 可以使用 **DB2_NUM_CKPW_DAEMONS** 注册表变量来启动可配置数目的检查密码守护程序。这些守护程序是在 db2start 期间创建的，当使用缺省安全插件 IBMOSauthserver 时，这些守护程序将处理检查密码请求。增大 **DB2_NUM_CKPW_DAEMONS** 的设置可以缩短建立数据库连接所需要的时间，但是此操作仅在下列情况下有效: 同时建立了许多连接，并且进行认证需要的成本较高。

  可将 **DB2_NUM_CKPW_DAEMONS** 设置为 1 到 100 之间的值。数据库管理器将按 **DB2_NUM_CKPW_DAEMONS** 指定的个数创建守护程序。每个守护程序都可以直接处理检查密码请求。

  可以添加可选 FORK 参数以使检查密码守护程序能够显式衍生用于处理检查密码请求的外部检查密码程序（db2ckpw）。这与前发行版中将 **DB2_NUM_CKPW_DAEMONS** 设置为 0 相似。在 FORK 方式中，每个检查密码守护程序会为每个检查密码请求衍生检查密码程序。FORK 方式的守护程序作为实例所有者启动。

  如果 **DB2_NUM_CKPW_DAEMONS** 设置为 0，那么有效的值为 3:FORK，表示将以 FORK 方式启动 3 个检查密码守护程序。

**DB2_OPTSTATS_LOG**

- 操作系统: 所有操作系统
- 缺省值: 未设置（请参阅下面的详细信息），值 = OFF 或 ON {NUM | SIZE | NAME | DIR}
- **DB2_OPTSTATS_LOG** 指定统计信息事件日志记录文件的属性，这些文件用于监视和分析与统计信息收集相关的活动。**DB2_OPTSTATS_LOG** 未设置或设置为 ON 时，将启用统计信息事件日志记录，从而允许您监视系统性能并保留历史记录以便更好地确定问题。日志记录将写入第一个日志文件中，直到该文件已满为止。后续记录将写入下一个可用的日志文件。如果达到文件的最大数目，那么将用新记录覆盖存在时间最长的日志文件。如果您非常担心系统资源消耗，请通过将此注册表变量设置为 OFF 来将其禁用。

显式启用统计信息事件日志记录时（设置为 ON），您可以修改一些选项:

- NUM: 旋转日志文件的最大数目。缺省值 = 5; 值: 1 - 15
- SIZE: 旋转日志文件的最大大小。（每个旋转文件的大小为 SIZE/NUM。）缺省值 = 100 Mb; 值: 1 Mb – 4096 Mb
- NAME: 旋转日志文件的基本名称。缺省值 = db2optstats.*number*.log，对于实例，它为 db2optstats.0.log、db2optstats.1.log 等。

- DIR: 旋转日志文件的基本目录。缺省值 = **$diagpath**/events

可以为任意多个这些选项指定值，但要启用统计信息日志记录时，应确保 ON 是第一个值。例如，要启用最大日志文件数为 6、最大文件大小为 25 Mb、基本文件名为 mystatslog 且目录为 mystats 的统计信息日志记录，请发出以下命令:

```
db2set DB2_OPTSTATS_LOG=ON,NUM=6,SIZE=25,NAME=mystatslog,DIR=mystats
```

### DB2REMOTEPREG

- 操作系统: Windows
- 缺省值: NULL; 值: 任何有效 Windows 机器名
- 指定包含 DB2 实例概要文件和 DB2 实例的 Win32 注册表列表的远程机器名。**DB2REMOTEPREG** 的值只应在安装 DB2 之后设置一次，且不应该修改。使用此变量要格外小心。

### DB2_RESOLVE_CALL_ CONFLICT

- 操作系统: AIX、HP-UX、Solaris、Linux 和 Windows
- 缺省值: YES; 值: YES、NO 或 ALL
- 在以下两种情况下发出 CALL 语句时将消除 SQLCODE SQL0746 运行时错误:
  - 在触发器中，将 **DB2_RESOLVE_CALL_CONFLICT** 设置为 YES（缺省值）或 ALL 时
  - 在 SQL 表函数中，将 **DB2_RESOLVE_CALL_CONFLICT** 设置为 ALL 时

  根据 SQL 标准有关执行顺序的规则，在完成由触发操作导致的修改之前，数据库管理器可能无法读取或修改触发器的主题表。在调用语句中的某个位置访问表函数的主题表时，数据库管理器可能也无法读取或修改该主题表。当多个表请求对相同调用语句中的这些主题表进行访问时，因为在触发器或 SQL 表函数尝试引用这些表时，这些表可能会更改，所以这些冲突的请求被称为*变异表冲突*。

  将此变量设置为 YES（缺省值）时，数据库管理器将根据需要通过使用临时表来严格实施特定执行顺序，在触发器中调用的过程中将遵循该执行顺序来访问表。

  将此变量设置为 ALL 时，数据库管理器将根据需要通过使用临时表来严格实施特定执行顺序，在触发器中调用的过程中以及在表函数中将遵循该执行顺序来访问表。从修订包 2 开始，可使用 ALL 设置。

  将此变量设置为 NO 时，数据库管理器假定不应该出现变异表冲突，但如果出现变异表冲突，那么数据库管理器将阻止变异表冲突并生成 SQLCODE SQL0746 错误消息。
- 在更改 **DB2_RESOLVE_CALL_CONFLICT** 的值之前，请停止该实例。在更改该值之后，请重新启动该实例，然后重新绑定任何导致调用触发器或表函数的包。要重新绑定 SQL 过程，请使用以下语句:

```
CALL SYSPROC.REBIND_ROUTINE_PACKAGE ('P','procedureschema.procedurename','CONSERVATIVE');
```

- **DB2_RESOLVE_CALL_CONFLICT** 在以下情况下会影响性能:

– 在触发器上，将该注册表变量设置为 YES 或 ALL 时。通常，此影响很小。因为在大多数情况下，触发语句仅修改一行或少数行，所以 OLTP 环境中的性能通常不受影响。遵循一般建议将系统管理的空间（SMS）用作临时表空间时，使用 **DB2_RESOLVE_CALL_CONFLICT** 所带来的性能影响可能很小。
– 在 SQL 表函数上，将该注册表变量设置为 ALL 时。影响程度取决于冲突类型:
  - 当 CALL 语句以及同一 SQL 表函数中的另一语句尝试访问同一表时，影响很小。
  - 当 SQL 表函数中的 CALL 语句以及对同一 SQL 表函数进行调用的 SELECT 语句尝试访问同一表时，影响程度取决于插入临时表中的数据量。

**DB2ROUTINE_DEBUG**

- 操作系统: AIX 和 Windows
- 缺省值: OFF; 值: ON 或 OFF
- 指定是否对 Java 存储过程启用调试功能。如果不调试 Java 存储过程，那么使用缺省值 OFF。启用调试对性能有影响。

**注**: 不推荐使用 **DB2ROUTINE_DEBUG**，将来的发行版中会将其除去。此存储过程调试器已替换为统一调试器。

**DB2SATELLITEID**

- 操作系统: 所有操作系统
- 缺省值: NULL; 值: 在卫星控制数据库中声明的有效卫星标识
- 指定在卫星进行同步时被传递至卫星控制服务器的卫星标识。如果没有为此变量指定值，那么将登录标识用作卫星标识。

**DB2_SERVER_CONTIMEOUT**

- 操作系统: 所有操作系统
- 缺省值: 180; 值: 0 到 32767 秒
- 此注册表变量和 **DB2_DISPATCHER_PEEKTIMEOUT** 注册表变量都配置在连接期间处理新客户机的方式。**DB2_SERVER_CONTIMEOUT** 允许您调整代理程序等待客户机的连接请求直至终止连接的时间，该时间以秒计。在大多数情况下，您无需调整此注册表变量，但如果在连接时 DB2 客户机常常被服务器作超时处理，您可以将 **DB2_SERVER_CONTIMEOUT** 设置为更高的值以延长超时时间。如果设置了无效值，那么将使用缺省值。此注册表变量不是动态的。

**DB2_SERVER_ENCALG**

- 操作系统: 所有操作系统
- 缺省值: NULL; 值: AES_CMP 或 AES_ONLY
- 当您连接至 DB2 Linux 版、UNIX 版和 Windows 版版本 9.5 修订包 3 和更高版本的数据库服务器时，可以使用“高级加密标准”（AES）对用户标识和密码进行加密。当数据库管理器配置参数 **authentication** 设置为 SERVER_ENCRYPT 时，不管客户机建议使用哪种加密算法，数据库服务器的缺省行为都是接受。可以使用注册表变量 **DB2_SERVER_ENCALG** 来更改此行为。它将影响与 DB2 实例的所有连接，与应用程序使用的语言无关。

如果此变量设置为 AES_ONLY，那么数据库服务器将只接受使用 AES 加密的连接。如果客户机不支持 AES 加密，那么连接将被拒绝。

如果此变量设置为 AES_CMP，那么数据库服务器将接受使用 AES 或 DES 进行加密的用户标识和密码。但是，如果客户机支持 AES 加密，那么它将针对 AES 进行协商。

如果通过程序使应用程序能够使用 AES 加密，那么不需要设置 **DB2_SERVER_ENCALG**。

**DB2SORT**

- 操作系统：所有操作系统，仅适用于服务器
- 缺省值：NULL
- 此变量指定 LOAD 实用程序在运行时要装入的库的位置。该库包含在对索引数据排序时使用的函数的入口点。使用 **DB2SORT** 来利用供应商提供的排序产品，以便在生产表索引时与 LOAD 实用程序一起使用。提供的路径必须是相对于数据库服务器的路径。

**DB2_THREAD_SUSPENSION**

- 操作系统：具有存储密钥支持的 AIX
- 缺省值：OFF；值：ON 或 OFF
- 此注册表变量启用或禁用 DB2 线程暂挂功能。它允许您通过暂挂出现问题的引擎线程（尝试非法访问受存储密钥保护的内存的线程）来控制 DB2 实例是否仍然落入陷阱。

  **注**：仅当 **DB2_MEMORY_PROTECT** 注册表变量设置为 YES 时，才能启用 **DB2_THREAD_SUSPENSION**。

**DB2_TRUNCATE_ REUSESTORAGE**

- 操作系统：所有操作系统
- 缺省值：NULL（未设置）；值：IMPORT 或 import
- 可以使用此变量来解决带有 REPLACE 命令的 IMPORT 与 BACKUP ... ONLINE 命令之间的锁定争用情况。在某些情况下，联机备份和截断操作无法同时执行。发生这种情况时，可以将 **DB2_TRUNCATE_REUSESTORAGE** 设置为 IMPORT 或 import，这将跳过对象（包括数据、索引、长字段、大对象和块映射（对于多维集群表））的物理截断，而是只执行逻辑截断。即，带有 REPLACE 命令的 IMPORT 将清空该表，从而导致该对象的逻辑大小减小，但仍分配磁盘存储器。

  这个注册表变量是动态的：可以设置或取消设置它，而不必停止并启动实例。可以在联机备份启动前设置 **DB2_TRUNCATE_REUSESTORAGE**，然后在联机备份完成后将其取消设置。对于多分区环境来说，此注册表变量只在设置了此变量的节点上处于活动状态。
  **DB2_TRUNCATE_REUSESTORAGE** 只对 DMS 永久对象有效。

  在 SAP 环境中，当设置了 **DB2_WORKLOAD**=SAP 时，此注册表变量的缺省值为 IMPORT。

**DB2_USE_DB2JCCT2_ JROUTINE**

- 操作系统：所有操作系统

- 缺省值：未设置；值：ON/YES/1/TRUE 或 OFF/NO/0/FALSE
- 用于 Java 存储过程和用户定义的函数的缺省驱动程序是 IBM 数据服务器 JDBC 和 SQLJ 驱动程序。如果要使用不推荐使用的 DB2 JDBC 2 类驱动程序 Linux 版、UNIX 版和 Windows 版来处理 Java 例程的 SQL 请求，请将 **DB2_USE_DB2JCCT2_JROUTINE** 设置为 OFF、NO、0 或 FALSE 中的任意一个。

**DB2_UTIL_MSGPATH**

- 操作系统：所有操作系统
- 缺省值：*instanceName*/tmp 目录
- **DB2_UTIL_MSGPATH** 注册表变量与 SYSPROC.ADMIN_CMD 过程、SYSPROC.ADMIN_REMOVE_MSGS 过程和 SYSPROC.ADMIN_GET_MSGS UDF 配合使用。它应用于实例级。可以设置 **DB2_UTIL_MSGPATH** 来指示受防护用户标识在服务器上可以读写和删除文件的目录路径。必须能从所有协调程序分区访问此目录，而且该目录必须有足够的空间来容纳实用程序消息文件。

  如果未设置此路径，那么使用 *instanceName*/tmp 目录作为缺省目录（请注意，卸载 DB2 时，将清除 *instanceName*/tmp）。

  更改此路径时，不会自动移动或删除先前设置所指向目录中的现有文件。如果要检索在旧路径下创建的消息内容，那么必须手动将这些消息（它们带有实用程序名前缀和用户标识后缀）移到 **DB2_UTIL_MSGPATH** 指向的新目录。以后，将在新位置中创建、读取和清除实用程序消息文件。

  **DB2_UTIL_MSGPATH** 目录下的文件都是特定于实用程序的，而与事务无关。它们不是备份映像的一部分。**DB2_UTIL_MSGPATH** 目录下的文件由用户管理；这表示用户可以使用 SYSPROC.ADMIN_REMOVE_UTILMSG 过程来删除消息文件。卸载 DB2 时不会清除这些文件。

**DB2_VENDOR_INI**

- 操作系统：AIX、HP-UX、Solaris 和 Windows
- 缺省值：NULL；值：任何有效路径和文件。
- 指向包含所有特定于供应商的环境设置的文件。当数据库管理器启动时读取该值。

**注**：在版本 9.5 中，不推荐使用 **DB2_VENDOR_INI**，将来的发行版中可能会将它除去。但是，可以将它包含的环境变量设置放入由 **DB2_DJ_INI** 变量指定的文件中。

**DB2_XBSA_LIBRARY**

- 操作系统：AIX、HP-UX、Solaris 和 Windows
- 缺省值：NULL；值：任何有效路径和文件。
- 指向供应商提供的 XBSA 库。在 AIX 上，如果共享对象未命名为 shr.o，设置必须包括该共享对象。HP-UX、Solaris 和 Windows 不需要共享对象名。例如，要对 DB2 使用 Legato 的 NetWorker Business Suite Module，必须按如下所示设置注册表变量：

  ```
  db2set DB2_XSBA_LIBRARY="/usr/lib/libxdb2.a(bsashr10.o)"
  ```

通过 BACKUP DATABASE 或 RESTORE DATABASE 命令可以调用 XBSA 接口。例如:

```
db2 backup db sample use XBSA
db2 restore db sample use XBSA
```

# 第 20 章 配置参数

当创建了 DB2 数据库实例或数据库时，就会使用缺省参数值创建相应的配置文件。可以修改这些参数值以提高性能以及实例或数据库的其他特征。

数据库管理器根据这些参数的缺省值分配的磁盘空间和内存可能足以满足您的需要。但在某些情况下，使用这些缺省值可能无法实现最佳性能。

*配置文件*包含一些参数，这些参数定义诸如分配给 DB2 数据库产品和各个数据库的资源以及诊断级别之类的值。有两种类型的配置文件:

- 每个 DB2 实例的数据库管理器配置文件
- 每个独立的数据库的数据库配置文件

*数据库管理器配置文件*是在创建 DB2 实例时创建的。它包含的参数在实例级影响系统资源，并且与该实例包含的任何一个数据库无关。根据系统的配置，可将这些参数中许多参数的值更改为非系统缺省值，以提高性能或增大容量。

每个客户机安装也有一个数据库管理器配置文件。此文件包含关于特定工作站的客户机启用程序的信息。在可用于服务器的参数中，有一个子集可用于客户机。

数据库管理器配置参数存储在名为 db2systm 的文件中。此文件是在创建数据库管理器的实例时创建的。在 Linux 和 UNIX 环境中，可以在数据库管理器的实例的 sqllib 子目录中找到此文件。在 Windows 中，此文件的缺省位置是 sqllib 目录的 instance 子目录。如果设置了 DB2INSTPROF 变量，那么文件在 DB2INSTPROF 变量指定的目录的 instance 子目录中。

在版本 9.5 中，全局级 DB2INSTPROF 的隐式缺省值 (db2set -g) 将存储在下面显示的新位置中，即使未设置 DB2INSTPROF 亦如此:

- 在 Vista 环境中: ProgramData\IBM\DB2\<DB2COPYNAME>
- 在 Windows 2003/XP 环境中: Documents and Settings\All Users\Application Data\IBM\DB2\<Copy Name>

指定运行时数据文件的存储位置的其他概要文件注册表变量应查询 DB2INSTPROF 的值。这包括下列变量:

- DB2CLINIPATH
- DIAGPATH
- SPM_LOG_PATH

对于在 V8.2 之前创建的数据库，数据库配置参数存储在一个名为 SQLDBCON 的文件中；而对于在 V8.2 和更高版本中创建的数据库，所有数据库配置参数都存储在一个名为 SQLDBCONF 的文件中。不能直接编辑这些文件，而只能通过提供的 API 或调用该 API 的工具来更改或查看。

在分区数据库环境中，此文件位于共享文件系统中，以便所有数据库分区服务器对同一文件都具有访问权。数据库管理器的配置在所有数据库分区服务器上都相同。

© Copyright IBM Corp. 1993, 2010

大多数参数会影响将分配给数据库管理器的单个实例的系统资源量，或者这些参数会配置数据库管理器和基于环境考虑的不同通信子系统的设置。另外，存在仅供参考的其他参数，不能更改这些参数。所有这些参数都具有全局适用性，独立于存储在数据库管理器的该实例下的任何单个数据库。

*数据库配置文件*是在创建数据库时创建的，它位于数据库所在的地方。每个数据库都有一个配置文件。其参数指定要分配给该数据库的资源量以及其他事项。您可以更改许多参数的值以提高性能或增大容量。根据特定数据库中活动类型的不同，可能需要进行不同的更改。

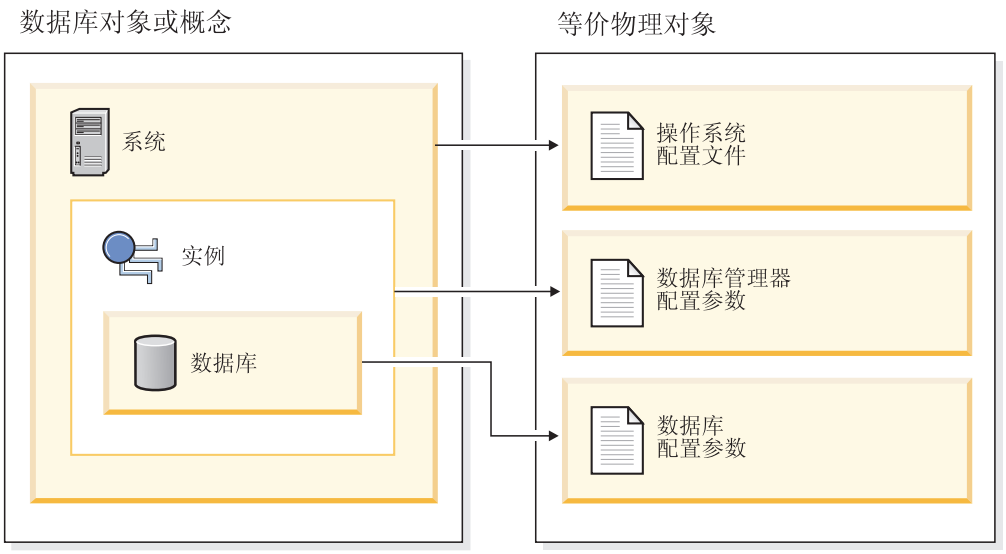



*图 29. 数据库对象与配置文件之间的关系*

## 使用配置参数配置 DB2 数据库管理器

数据库管理器根据这些参数的缺省值分配的磁盘空间和内存可能足以满足您的需要。但在某些情况下，使用这些缺省值可能无法实现最佳性能。

### 关于本任务

因为缺省值适用于内存资源相对较少且专门用作数据库服务器的机器，所以如果您的环境存在以下情况，那么可能需要修改这些值:

- 大型数据库
- 大量连接
- 对特定应用程序有高性能要求
- 唯一的查询或事务负载或类型

在一个或多个方面，每个事务处理环境都唯一。当使用缺省配置时，这些差异可能对数据库管理器的性能产生深远的影响。因此，强烈建议您调整您的环境配置。

调整配置的较好起点是使用“配置顾问程序”或 AUTOCONFIGURE 命令，它们将根据您对工作负载特征问题的回答来生成参数值。

可以将某些配置参数设置为 AUTOMATIC，从而允许 DB2 确定要用于该参数的值。当配置参数的值设置为 AUTOMATIC 时，某些配置参数将监视和调整正在服务器上运行的工作负载，尤其是与自调整内存相关联的配置参数将调整正在运行的工作负载。如果其他配置参数的值为 AUTOMATIC，这意味着在激活实例或数据库时作出决定。要关闭配置参数的 AUTOMATIC 设置并同时维护当前内部设置，请将 **MANUAL** 关键字与 UPDATE DATABASE CONFIGURATION 命令配合使用。如果数据库管理器更新这些参数的值，那么 GET DB CFG SHOW DETAIL 和 GET DBM CFG SHOW DETAIL 命令将显示新值。

个别数据库的参数存储在名为 SQLDBCONF 的配置文件中。此文件与 SQL*nnnnn* 目录中的数据库的其他控制文件存储在一起，其中 *nnnnn* 是创建数据库时指定的数字。每个数据库都有它自己的配置文件，并且文件中的大多数参数指定分配给该数据库的资源量。该文件还包含描述信息以及指示数据库的状态的标志。

**注意**：如果您使用非数据库管理器提供的方法来编辑 db2systm、SQLDBCON 或 SQLDBCONF，那么可能会使数据库不可用。不要使用非数据库管理器记载和支持的方法来更改这些文件。

在分区数据库环境中，对于每个数据库分区都存在一个单独的 SQLDBCONF 文件。在每个数据库分区上，SQLDBCONF 文件中的值可能相同或不同，但是在同机种环境中，建议配置参数值在所有数据库分区上都相同。通常，可能有一个目录节点需要不同的数据库配置参数设置，而其他数据分区也具有不同的值，这取决于它们的机器类型和其他信息。

### 过程

1. 更新配置参数。
   - 使用命令行处理器来更新:

     可以按如下所示输入用于更改这些设置的命令:

     对于数据库管理器配置参数:
     - GET DATABASE MANAGER CONFIGURATION（或者 GET DBM CFG）
     - UPDATE DATABASE MANAGER CONFIGURATION（或者 UPDATE DBM CFG）
     - RESET DATABASE MANAGER CONFIGURATION（或者 RESET DBM CFG），以便将*所有*数据库管理器参数复位为其缺省值
     - AUTOCONFIGURE

     对于数据库配置参数:
     - GET DATABASE CONFIGURATION（或者 GET DB CFG）
     - UPDATE DATABASE CONFIGURATION（或者 UPDATE DB CFG）
     - RESET DATABASE CONFIGURATION（或者 RESET DB CFG），以便将*所有*数据库参数复位为其缺省值
     - AUTOCONFIGURE
   - 使用应用程序编程接口 (API) 来更新:

     API 可从应用程序或主语言程序中调用。调用以下 DB2 API 来查看或更新配置参数:

- db2AutoConfig - 访问配置顾问程序
- db2CfgGet - 获取数据库管理器参数或数据库配置参数
- db2CfgSet - 设置数据库管理器参数或数据库配置参数

• 使用配置助手来更新

还可以使用“配置助手”在客户机上设置数据库管理器配置参数。可以联机更改其他参数；这些参数称为*可配置的联机配置参数*。

2. 查看已更新的配置值。

对于某些数据库管理器配置参数，必须停止数据库管理器 (db2stop)，然后重新启动它 (db2start) 才能使新的参数值生效。

对于某些数据库参数，仅当重新激活数据库或者数据库从脱机状态转变为联机状态时，更改才将生效。在这些情况下，所有应用程序必须首先与该数据库断开连接。（如果已激活该数据库，或者它已从脱机状态转变为联机状态，那么必须先取消激活，然后再重新激活。）然后，当与该数据库建立第一个新的连接时，这些更改才生效。

如果在连接至某个实例时更改可配置的联机数据库管理器配置参数的设置，那么 UPDATE DBM CFG 命令的缺省行为将是立即应用更改。如果您不想立即应用更改，那么在 UPDATE DBM CFG 命令上使用 **DEFERRED** 选项。

要联机更改数据库管理器配置参数:

```
db2 attach to instance-name
db2 update dbm cfg using parameter-name value
   db2 detach
```

对于客户机，数据库管理器配置参数的更改在下次该客户机与服务器连接时生效。

如果在连接时更改可配置联机数据库配置参数，那么缺省行为是联机应用更改（只要有可能）。您应该注意，某些参数更改由于与分配空间相关的开销而可能花相当长的时间才会生效。要从命令行处理器联机更改配置参数，需要与数据库的连接。要联机更改数据库配置参数:

```
db2 connect to dbname
db2 update db cfg using parameter-name parameter-value
   db2 connect reset
```

每个可配置的联机配置参数都有一个与之关联的*传播类*。传播类指示您可以期望配置参数的更改何时生效。有四种传播类:

- **立即**: 在命令或 API 调用中立即更改的参数。例如，**diaglevel** 具有立即传播类。
- **语句边界**: 在语句或类似语句的边界上更改的参数。例如，如果您更改 **sortheap** 的值，那么所有新的请求都将使用该新值。
- **事务边界**: 在事务边界上更改的参数。例如，在 COMMIT 语句之后更新 **dl_expint** 的新值。
- **连接**: 与数据库建立新连接时更改的参数。例如，**dft_degree** 的新值对于连接至数据库的新应用程序生效。

当新参数值可能未立即生效时，查看参数设置（使用 GET DATABASE MANAGER CONFIGURATION 或 GET DATABASE CONFIGURATION 命令）将始终显示最新的更新。使用这些命令上的 **SHOW DETAIL** 子句查看参数设置将显示内存中的最新更新和值。

3. 更新数据库配置参数之后重新绑定应用程序。

   更改某些数据库配置参数可能会影响由 SQL 和 XQuery 优化器选择的访问方案。在更改其中任何一个参数后，应考虑重新绑定应用程序来确保对 SQL 和 XQuery 语句使用最佳的访问方案。任何联机修改的参数（例如，通过使用 UPDATE DATABASE CONFIGURATION IMMEDIATE 命令）都将导致 SQL 和 XQuery 优化器为新查询语句选择一个新的访问方案。但查询语句高速缓存不会清除现有条目。要清除查询高速缓存的内容，可使用 FLUSH PACKAGE CACHE 语句。

   **注**: 在帮助文档和其他 DB2 文档中，许多配置参数（例如，**userexit**）被描述为具有可接受的值 Yes 或 No，或者具有 On 或 Off。为了清楚起见，应将 Yes 视为等价于 On，而应将 No 视为等价于 Off。

# 配置参数摘要

下表列示了数据库服务器的数据库管理器和数据库配置文件中的参数。更改数据库管理器和数据库配置参数时，要考虑每个参数的详细信息。包括缺省值的特定操作环境信息是每个参数描述的一部分。

## 数据库管理器配置参数摘要

对于某些数据库管理器配置参数，必须停止数据库管理器（db2stop），然后重新启动它（db2start）才能使新的参数值生效。可以联机更改其他参数；这些参数称为*可配置的联机配置参数*。如果在连接至某个实例时更改可配置的联机数据库管理器配置参数的设置，那么 UPDATE DBM CFG 命令的缺省行为会立即应用更改。如果您不想立即应用更改，那么在 UPDATE DBM CFG 命令上使用 **DEFERRED** 选项。

下表中的“自动”列指示参数是否支持 UPDATE DBM CFG 命令中的 **AUTOMATIC** 关键字。

将某个参数更新为 AUTOMATIC 时，还可以指定起始值和 **AUTOMATIC** 关键字。请注意，值对每个参数可以具有不同的含义，并且在某些情况下不适用。在指定值之前，请阅读参数的文档以确定它所表示的含义。在以下示例中，**num_poolagents** 将更新为 AUTOMATIC 并且数据库管理器将使用 20 作为要合用的空闲代理程序的最小数目:

```
db2 update dbm cfg using num_poolagents 20 automatic
```

要复位 AUTOMATIC 功能，可以将此参数更新为一个值或者可以使用 **MANUAL** 关键字。将某个参数更新为 MANUAL 时，此参数不再是自动的，并且会被设置为其当前值（如 GET DBM CFG SHOW DETAIL 或 GET DB CFG SHOW DETAIL 命令生成的 `Current Value` 列中所显示的那样）。

“性能影响”列指示每个参数影响系统性能的相对程度。不可能将此列准确地应用于所有环境；您应该将此信息视为一般情况。

- **高** - 指示该参数会对性能产生重大影响。应有意识地决定这些参数的值，在某些情况下，这意味着将接受所提供的缺省值。

- **中** - 指示该参数会对性能产生某些影响。您的特定环境和需要将确定应对这些参数进行多大程度的调整。
- **低** - 指示该参数会对性能产生不是很普遍或者不太重要的影响。
- **无** - 指示该参数对性能没有直接影响。尽管不必调整这些参数来增强性能，但是它们对于系统配置的其他方面（例如，通信支持）可能很重要。

“标记”、“标记值”和“数据类型”列提供调用 db2CfgGet 或 db2CfgSet API 时将需要的信息。此信息包括配置参数标识、db2CfgParam 数据结构中的 **token** 元素的条目和传递至该结构的值的数据类型。

*表 65. 可配置的数据库管理器配置参数*

| 参数 | 可联机配置 | 自动 | 性能影响 | 标记 | 标记值 | 数据类型 | 其他信息 |
|---|---|---|---|---|---|---|---|
| **agent_stack_sz** | 否 | 否 | 低 | SQLF_KTN_AGENT_STACK_SZ | 61 | Uint16 | 第 426 页的『agent_stack_sz - 代理程序堆栈大小』 |
| **agentpri** | 否 | 否 | 高 | SQLF_KTN_AGENTPRI | 26 | Sint16 | 第 428 页的『agentpri - 代理程序的优先级』 |
| **aslheapsz** | 否 | 否 | 高 | SQLF_KTN_ASLHEAPSZ | 15 | Uint32 | 第 429 页的『aslheapsz - 应用程序支持层堆大小』 |
| **audit_buf_sz** | 否 | 否 | 高 | SQLF_KTN_AUDIT_BUF_SZ | 312 | Sint32 | 第 430 页的『audit_buf_sz - 审计缓冲区大小』 |
| **authentication**[1] | 否 | 否 | 低 | SQLF_KTN_AUTHENTICATION | 78 | Uint16 | 第 431 页的『authentication - 认证类型』 |
| **catalog_noauth** | 是 | 否 | 无 | SQLF_KTN_CATALOG_NOAUTH | 314 | Uint16 | 第 432 页的『catalog_noauth - 没有权限时允许的编目』 |
| **clnt_krb_plugin** | 否 | 否 | 无 | SQLF_KTN_CLNT_KRB_PLUGIN | 812 | char(33) | 第 433 页的『clnt_krb_plugin - 客户机 Kerberos 插件』 |
| **clnt_pw_plugin** | 否 | 否 | 无 | SQLF_KTN_CLNT_PW_PLUGIN | 811 | char(33) | 第 433 页的『clnt_pw_plugin - 客户机用户标识密码插件』 |
| **cluster_mgr** | 否 | 否 | 无 | SQLF_KTN_CLUSTER_MGR | 920 | char(262) | 第 433 页的『cluster_mgr - 集群管理器名称』 |
| **comm_bandwidth** | 是 | 否 | 中 | SQLF_KTN_COMM_BANDWIDTH | 307 | float | 第 434 页的『comm_bandwidth - 通信带宽』 |
| **conn_elapse** | 是 | 否 | 中 | SQLF_KTN_CONN_ELAPSE | 508 | Uint16 | 第 435 页的『conn_elapse - 连接耗用时间』 |
| **cpuspeed** | 是 | 否 | 高 | SQLF_KTN_CPUSPEED | 42 | float | 第 435 页的『cpuspeed - CPU 速度』 |
| **dft_account_str** | 是 | 否 | 无 | SQLF_KTN_DFT_ACCOUNT_STR | 28 | char(25) | 第 436 页的『dft_account_str - 缺省对方付费帐户』 |
| **dft_monswitches**<br>• **dft_mon_bufpool**<br>• **dft_mon_lock**<br>• **dft_mon_sort**<br>• **dft_mon_stmt**<br>• **dft_mon_table**<br>• **dft_mon_timestamp**<br>• **dft_mon_uow** | 是 | 否 | 中 | SQLF_KTN_DFT_MONSWITCHES[2]<br>• SQLF_KTN_DFT_MON_BUFPOOL<br>• SQLF_KTN_DFT_MON_LOCK<br>• SQLF_KTN_DFT_MON_SORT<br>• SQLF_KTN_DFT_MON_STMT<br>• SQLF_KTN_DFT_MON_TABLE<br>• SQLF_KTN_DFT_MON_TIMESTAMP<br>• SQLF_KTN_DFT_MON_UOW | 29<br>• 33<br>• 34<br>• 35<br>• 31<br>• 32<br>• 36<br>• 30 | Uint16<br>• Uint16<br>• Uint16<br>• Uint16<br>• Uint16<br>• Uint16<br>• Uint16<br>• Uint16 | 第 436 页的『dft_monswitches - 缺省数据库系统监视器开关』 |
| **dftdbpath** | 是 | 否 | 无 | SQLF_KTN_DFTDBPATH | 27 | char(215) | 第 437 页的『dftdbpath - 缺省数据库路径』 |

*表 65. 可配置的数据库管理器配置参数 (续)*

| 参数 | 可联机配置 | 自动 | 性能影响 | 标记 | 标记值 | 数据类型 | 其他信息 |
|---|---|---|---|---|---|---|---|
| **diaglevel** | 是 | 否 | 低 | SQLF_KTN_DIAGLEVEL | 64 | Uint16 | 第 438 页的『diaglevel -"诊断错误捕获级别"』 |
| **diagpath** | 是 | 否 | 无 | SQLF_KTN_DIAGPATH | 65 | char(215) | 第 439 页的『diagpath -"诊断数据目录路径"』 |
| **dir_cache** | 否 | 否 | 中 | SQLF_KTN_DIR_CACHE | 40 | Uint16 | 第 442 页的『dir_cache - 目录高速缓存支持』 |
| **discover**[3] | 否 | 否 | 中 | SQLF_KTN_DISCOVER | 304 | Uint16 | 第 443 页的『discover - 发现方式』 |
| **discover_inst** | 是 | 否 | 低 | SQLF_KTN_DISCOVER_INST | 308 | Uint16 | 第 443 页的『discover_inst - 发现服务器实例』 |
| **fcm_num_buffers** | 是 | 是 | 中 | SQLF_KTN_FCM_NUM_ BUFFERS | 503 | Uint32 | 第 444 页的『fcm_num_buffers - FCM 缓冲区数』 |
| **fcm_num_channels** | 是 | 是 | 中 | SQLF_KTN_FCM_NUM_ CHANNELS | 902 | Uint32 | 第 445 页的『fcm_num_channels - FCM 通道数』 |
| **fed_noauth** | 是 | 否 | 无 | SQLF_KTN_FED_NOAUTH | 806 | Uint16 | 第 446 页的『fed_noauth - 绕过联合认证』 |
| **联合** | 是 | 否 | 中 | SQLF_KTN_FEDERATED | 604 | Sint16 | 第 446 页的『federated - 联合数据库系统支持』 |
| **federated_async** | 是 | 是 | 中 | SQLF_KTN_FEDERATED_ ASYNC | 849 | Sint32 | 第 446 页的『federated_async - 每个查询的最大异步 TQ 数配置参数』 |
| **fenced_pool** | 是 | 是 | 中 | SQLF_KTN_FENCED_POOL | 80 | Sint32 | 第 447 页的『fenced_pool - 最大受防护进程数』 |
| **group_plugin** | 否 | 否 | 无 | SQLF_KTN_GROUP_PLUGIN | 810 | char(33) | 第 448 页的『group_plugin - 组插件』 |
| **health_mon** | 是 | 否 | 低 | SQLF_KTN_HEALTH_MON | 804 | Uint16 | 第 449 页的『health_mon - 运行状况监视器』 |
| **indexrec**[4] | 是 | 否 | 中 | SQLF_KTN_INDEXREC | 20 | Uint16 | 第 449 页的『indexrec - 索引重新创建时间』 |
| **instance_memory** | 是 | 是 | 中 | SQLF_KTN_INSTANCE_ MEMORY | 803 | Uint64 | 第 451 页的『instance_memory - 实例内存』 |
| **intra_parallel** | 否 | 否 | 高 | SQLF_KTN_INTRA_PARALLEL | 306 | Sint16 | 第 453 页的『intra_parallel - 启用分区内并行性』 |
| **java_heap_sz** | 否 | 否 | 高 | SQLF_KTN_JAVA_HEAP_SZ | 310 | Sint32 | 第 453 页的『java_heap_sz - 最大 Java 解释器堆大小』 |
| **jdk_path** | 否 | 否 | 无 | SQLF_KTN_JDK_PATH | 311 | char(255) | 第 454 页的『jdk_path - Java 软件开发者工具箱安装路径』 |
| **keepfenced** | 否 | 否 | 中 | SQLF_KTN_KEEPFENCED | 81 | Uint16 | 第 455 页的『keepfenced - 保留受防护进程』 |
| **local_gssplugin** | 否 | 否 | 无 | SQLF_KTN_LOCAL_ GSSPLUGIN | 816 | char(33) | 第 455 页的『local_gssplugin - 用于实例级本地授权的 GSS API 插件』 |
| **max_connections** | 是 | 是 | 中 | SQLF_KTN_MAX_ CONNECTIONS | 802 | Sint32 | 第 456 页的『max_connections - 最大客户机连接数』 |
| **max_connretries** | 是 | 否 | 中 | SQLF_KTN_MAX_ CONNRETRIES | 509 | Uint16 | 第 456 页的『max_connretries - 节点连接重试次数』 |
| **max_coordagents** | 是 | 是 | 中 | SQLF_KTN_MAX_ COORDAGENTS | 501 | Sint32 | 第 457 页的『max_coordagents - 最大协调代理程序数』 |
| **max_querydegree** | 是 | 否 | 高 | SQLF_KTN_MAX_ QUERYDEGREE | 303 | Sint32 | 第 458 页的『max_querydegree - 最大查询并行度』 |
| **max_time_diff** | 否 | 否 | 中 | SQLF_KTN_MAX_TIME_DIFF | 510 | Uint16 | 第 458 页的『max_time_diff - 节点间的最大时差』 |
| **mon_heap_sz** | 是 | 是 | 低 | SQLF_KTN_MON_HEAP_SZ | 79 | Uint16 | 第 460 页的『mon_heap_sz -"数据库系统监视器堆大小"』 |

*表 65. 可配置的数据库管理器配置参数 (续)*

| 参数 | 可联机配置 | 自动 | 性能影响 | 标记 | 标记值 | 数据类型 | 其他信息 |
|---|---|---|---|---|---|---|---|
| **notifylevel** | 是 | 否 | 低 | SQLF_KTN_NOTIFYLEVEL | 605 | Sint16 | 第 462 页的『notifylevel - 通知级别』 |
| **num_initagents** | 否 | 否 | 中 | SQLF_KTN_NUM_ INITAGENTS | 500 | Uint32 | 第 463 页的『num_initagents - 池中的初始代理程序数』 |
| **num_initfenced** | 否 | 否 | 中 | SQLF_KTN_NUM_ INITFENCED | 601 | Sint32 | 第 463 页的『num_initfenced - 受防护进程的初始数目』 |
| **num_poolagents** | 是 | 是 | 高 | SQLF_KTN_NUM_ POOLAGENTS | 502 | Sint32 | 第 464 页的『num_poolagents - 代理程序池大小』 |
| **numdb** | 否 | 否 | 低 | SQLF_KTN_NUMDB | 6 | Uint16 | 第 465 页的『numdb - 同时处于活动状态的数据库（包括主机和 System i 数据库）的最大数目』 |
| **query_heap_sz** | 否 | 否 | 中 | SQLF_KTN_QUERY_HEAP_SZ | 49 | Sint32 | 第 465 页的『query_heap_sz - 查询堆大小』 |
| **resync_interval** | 否 | 否 | 无 | SQLF_KTN_RESYNC_ INTERVAL | 68 | Uint16 | 第 467 页的『resync_interval - 事务再同步时间间隔』 |
| **rqrioblk** | 否 | 否 | 高 | SQLF_KTN_RQRIOBLK | 1 | Uint16 | 第 467 页的『rqrioblk - 客户机 I/O 块大小』 |
| **sheapthres** | 否 | 否 | 高 | SQLF_KTN_SHEAPTHRES | 21 | Uint32 | 第 468 页的『sheapthres -"排序堆阈值"』 |
| **spm_log_file_sz** | 否 | 否 | 低 | SQLF_KTN_SPM_LOG_FILE_ SZ | 90 | Sint32 | 第 469 页的『spm_log_file_sz - 同步点管理器日志文件大小』 |
| **spm_log_path** | 否 | 否 | 中 | SQLF_KTN_SPM_LOG_PATH | 313 | char(226) | 第 470 页的『spm_log_path - 同步点管理器日志文件路径』 |
| **spm_max_resync** | 否 | 否 | 低 | SQLF_KTN_SPM_MAX_ RESYNC | 91 | Sint32 | 第 470 页的『spm_max_resync - 同步点管理器再同步代理程序限制』 |
| **spm_name** | 否 | 否 | 无 | SQLF_KTN_SPM_NAME | 92 | char(8) | 第 471 页的『spm_name - 同步点管理器名』 |
| **srvcon_auth** | 否 | 否 | 无 | SQLF_KTN_SRVCON_AUTH | 815 | Uint16 | 第 471 页的『srvcon_auth - 用于服务器中的入局连接的认证类型』 |
| **srvcon_gssplugin_list** | 否 | 否 | 无 | SQLF_KTN_SRVCON_ GSSPLUGIN_ LIST | 814 | char(256) | 第 471 页的『srvcon_gssplugin_list - 用于服务器中的入局连接的 GSS API 插件列表』 |
| **srv_plugin_mode** | 否 | 否 | 无 | SQLF_KTN_SRV_PLUGIN_ MODE | 809 | Uint16 | 第 473 页的『srv_plugin_mode - 服务器插件方式』 |
| **srvcon_pw_plugin** | 否 | 否 | 无 | SQLF_KTN_SRVCON_PW_ PLUGIN | 813 | char(33) | 第 472 页的『srvcon_pw_plugin - 用于服务器中的入局连接的用户标识密码插件』 |
| **start_stop_time** | 是 | 否 | 低 | SQLF_KTN_START_STOP_ TIME | 511 | Uint16 | 第 473 页的『start_stop_time - 启动和停止超时』 |
| **svcename** | 否 | 否 | 无 | SQLF_KTN_SVCENAME | 24 | char(14) | 第 474 页的『svcename - TCP/IP 服务名称』 |
| **sysadm_group** | 否 | 否 | 无 | SQLF_KTN_SYSADM_ GROUP | 39 | char(128) | 第 474 页的『sysadm_group - 系统管理权限组名』 |
| **sysctrl_group** | 否 | 否 | 无 | SQLF_KTN_SYSCTRL_ GROUP | 63 | char(128) | 第 475 页的『sysctrl_group - 系统控制权限组名』 |
| **sysmaint_group** | 否 | 否 | 无 | SQLF_KTN_SYSMAINT_ GROUP | 62 | char(128) | 第 475 页的『sysmaint_group - 系统维护权限组名』 |
| **sysmon_group** | 否 | 否 | 无 | SQLF_KTN_SYSMON GROUP | 808 | char(128) | 第 476 页的『sysmon_group - 系统监视权限组名』 |
| **tm_database** | 否 | 否 | 无 | SQLF_KTN_TM_DATABASE | 67 | char(8) | 第 477 页的『tm_database - 事务管理器数据库名称』 |

*表 65. 可配置的数据库管理器配置参数 (续)*

| 参数 | 可联机配置 | 自动 | 性能影响 | 标记 | 标记值 | 数据类型 | 其他信息 |
|---|---|---|---|---|---|---|---|
| **tp_mon_name** | 否 | 否 | 无 | SQLF_KTN_TP_MON_NAME | 66 | char(19) | 第 477 页的『tp_mon_name - 事务处理器监视器名』 |
| **trust_allclnts**[5] | 否 | 否 | 无 | SQLF_KTN_TRUST_ALLCLNTS | 301 | Uint16 | 第 479 页的『trust_allclnts - 信赖所有客户机』 |
| **trust_clntauth** | 否 | 否 | 无 | SQLF_KTN_TRUST_CLNTAUTH | 302 | Uint16 | 第 479 页的『trust_clntauth - 可信客户机认证』 |
| **util_impact_lim** | 是 | 否 | 高 | SQLF_KTN_UTIL_IMPACT_LIM | 807 | Uint32 | 第 480 页的『util_impact_lim - 实例影响策略』 |

**注:**

1. sqlenv.h 中定义了有效值。
2. 

```
Bit 1 (xxxx xxx1): dft_mon_uow
  Bit 2 (xxxx xx1x): dft_mon_stmt
  Bit 3 (xxxx x1xx): dft_mon_table
  Bit 4 (xxxx 1xxx): dft_mon_buffpool
  Bit 5 (xxx1 xxxx): dft_mon_lock
  Bit 6 (xx1x xxxx): dft_mon_sort
  Bit 7 (x1xx xxxx): dft_mon_timestamp
```

3. 有效值（在 sqlutil.h 中定义）为:

```
SQLF_DSCVR_KNOWN (1)
  SQLF_DSCVR_SEARCH (2)
```

4. 有效值（在 sqlutil.h 中定义）为:

```
SQLF_INX_REC_SYSTEM (0)
  SQLF_INX_REC_REFERENCE (1)
```

5. 有效值（在 sqlutil.h 中定义）为:

```
SQLF_TRUST_ALLCLNTS_NO (0)
  SQLF_TRUST_ALLCLNTS_YES (1)
  SQLF_TRUST_ALLCLNTS_DRDAONLY (2)
```

*表 66. 参考数据库管理器配置参数*

| 参数 | 标记 | 标记值 | 数据类型 | 其他信息 |
|---|---|---|---|---|
| **nodetype**[1] | SQLF_KTN_NODETYPE | 100 | Uint16 | 第 461 页的『nodetype - 机器节点类型』 |
| **release** | SQLF_KTN_RELEASE | 101 | Uint16 | 第 466 页的『release - 配置文件发行版级别』 |

**注:**

1. 有效值（在 sqlutil.h 中定义）为:

```
SQLF_NT_STANDALONE (0)
  SQLF_NT_SERVER (1)
  SQLF_NT_REQUESTOR (2)
  SQLF_NT_STAND_REQ (3)
  SQLF_NT_MPP (4)
  SQLF_NT_SATELLITE (5)
```

## 数据库配置参数摘要

下表列示了数据库配置文件中的参数。当更改该数据库配置参数时，要参考该参数的详细信息。

对于某些数据库配置参数，仅当重新激活数据库时，更改才将生效。在这些情况下，所有应用程序必须首先与该数据库断开连接。（如果已激活该数据库，那么必须先取消激活，然后再重新激活。）这些更改将在下次连接数据库时生效。可以联机更改其他参数；这些参数称为*可配置的联机配置参数*。

请参阅上面的『数据库管理器配置参数总结』一节，以了解“自动”、“性能影响”、“标记”、“标记值”和“数据类型”列的描述。

UPDATE DB CFG 命令中也支持 **AUTOMATIC** 关键字。在以下示例中，进一步更改此参数时，**database_memory** 将更新为 AUTOMATIC 并且数据库管理器将使用 20000 作为起始值。

```
db2 update db cfg using for sample using database_memory 20000 automatic
```

从版本 9.5 起，可以更新并复位一些或所有平台上的数据库配置参数值，而不必发出 db2_all 命令，或者不必单独地更新或复位每个分区。

*表 67. 可配置的数据库配置参数*

| 参数 | 可联机配置 | 自动 | 性能影响 | 标记 | 标记值 | 数据类型 | 其他信息 |
|---|---|---|---|---|---|---|---|
| **alt_collate** | 否 | 否 | 无 | SQLF_DBTN_ALT_COLLATE | 809 | Uint32 | 第 481 页的『alt_collate - 备用整理顺序』 |
| **applheapsz** | 是 | 是 | 中 | SQLF_DBTN_APPLHEAPSZ | 51 | Uint16 | 第 484 页的『applheapsz - 应用程序堆大小』 |
| **appl_memory** | 是 | 是 | 中 | SQLF_DBTN_APPL_MEMORY | 904 | Uint64 | 第 483 页的『appl_memory - 应用程序内存配置参数』 |
| **archretrydelay** | 是 | 否 | 无 | SQLF_DBTN_ ARCHRETRYDELAY | 828 | Uint16 | 第 485 页的『archretrydelay - 出错时的归档重试延迟』 |
| • **auto_maint**<br>• **auto_db_backup**<br>• **auto_tbl_maint**<br>• **auto_runstats**<br>• **auto_stats_prof**<br>• **auto_stmt_stats**<br>• **auto_prof_upd**<br>• **auto_reorg** | 是 | 否 | 中 | • SQLF_DBTN_AUTO_MAINT<br>• SQLF_DBTN_AUTO_DB_ BACKUP<br>• SQLF_DBTN_AUTO_TBL_ MAINT<br>• SQLF_DBTN_AUTO_ RUNSTATS<br>• SQLF_DBTN_AUTO_STATS_ PROF<br>• SQLF_DBTN_AUTO_STMT_ STATS<br>• SQLF_DBTN_AUTO_PROF_ UPD<br>• SQLF_DBTN_AUTO_REORG | • 831<br>• 833<br>• 835<br>• 837<br>• 839<br>• 905<br>• 844<br>• 841 | Uint16 | 第 485 页的『auto_maint - 自动维护』 |
| **auto_del_rec_obj** | 是 | 否 | 中 | SQLF_DBTN_AUTO_DEL_ REC_OBJ | 912 | Uint16 | 第 485 页的『auto_del_rec_obj - 自动删除恢复对象配置参数』 |
| **autorestart** | 是 | 否 | 低 | SQLF_DBTN_AUTO_RESTART | 25 | Uint16 | 第 487 页的『autorestart - 允许自动重新启动』 |
| **avg_appls** | 是 | 是 | 高 | SQLF_DBTN_AVG_APPLS | 47 | Uint16 | 第 488 页的『avg_appls - 平均活动应用程序数』 |
| **blk_log_dsk_ful** | 是 | 否 | 无 | SQLF_DBTN_BLK_LOG_DSK_ FUL | 804 | Uint16 | 第 489 页的『blk_log_dsk_ful - 磁盘已满时阻止进行日志记录』 |
| **blocknonlogged** | 是 | 否 | 低 | SQLF_DBTN_BLOCKNONLOGGED | 940 | Uint16 | 第 489 页的『blocknonlogged -“禁止创建允许不进行日志记录的活动的表”』 |
| **catalogcache_sz** | 是 | 否 | 中 | SQLF_DBTN_ CATALOGCACHE_SZ | 56 | Sint32 | 第 490 页的『catalogcache_sz - 目录高速缓存大小』 |
| **chngpgs_thresh** | 否 | 否 | 高 | SQLF_DBTN_CHNGPGS_ THRESH | 38 | Uint16 | 第 491 页的『chngpgs_thresh - 更改的页数阈值』 |
| **database_memory** | 是 | 是 | 中 | SQLF_DBTN_DATABASE_ MEMORY | 803 | Uint64 | 第 493 页的『database_memory - 数据库共享内存大小』 |
| **dbheap** | 是 | 是 | 中 | SQLF_DBTN_DB_HEAP | 58 | Uint64 | 第 496 页的『dbheap - 数据库堆』 |

*表 67. 可配置的数据库配置参数 (续)*

| 参数 | 可联机配置 | 自动 | 性能影响 | 标记 | 标记值 | 数据类型 | 其他信息 |
|---|---|---|---|---|---|---|---|
| **db_mem_thresh** | 是 | 否 | 低 | SQLF_DBTN_DB_MEM_THRESH | 849 | Uint16 | 第 495 页的『db_mem_thresh - 数据库内存阈值』 |
| **decflt_rounding** | 否 | 否 | 无 | SQLF_DBTN_DECFLT_ROUNDING | 913 | Unit16 | 第 497 页的『decflt_rounding - 十进制浮点数舍入配置参数』 |
| **dft_degree** | 是 | 否 | 高 | SQLF_DBTN_DFT_DEGREE | 301 | Sint32 | 第 498 页的『dft_degree - 缺省度』 |
| **dft_extent_sz** | 是 | 否 | 中 | SQLF_DBTN_DFT_EXTENT_SZ | 54 | Uint32 | 第 499 页的『dft_extent_sz - 表空间的缺省扩展数据块大小』 |
| **dft_loadrec_ses** | 是 | 否 | 中 | SQLF_DBTN_DFT_LOADREC_SES | 42 | Sint16 | 第 499 页的『dft_loadrec_ses - 缺省装入恢复会话数』 |
| **dft_mttb_types** | 否 | 否 | 无 | SQLF_DBTN_DFT_MTTB_TYPES | 843 | Uint32 | 第 500 页的『dft_mttb_types - 为优化缺省维护的表类型』 |
| **dft_prefetch_sz** | 是 | 是 | 中 | SQLF_DBTN_DFT_PREFETCH_SZ | 40 | Sint16 | 第 500 页的『dft_prefetch_sz - 缺省预取大小』 |
| **dft_queryopt** | 是 | 否 | 中 | SQLF_DBTN_DFT_QUERYOPT | 57 | Sint32 | 第 501 页的『dft_queryopt - 缺省查询优化类』 |
| **dft_refresh_age** | 否 | 否 | 中 | SQLF_DBTN_DFT_REFRESH_AGE | 702 | char(22) | 第 502 页的『dft_refresh_age - 缺省刷新持续时间』 |
| **dft_sqlmathwarn** | 否 | 否 | 无 | SQLF_DBTN_DFT_SQLMATHWARN | 309 | Sint16 | 第 502 页的『dft_sqlmathwarn - 出现算术异常时继续』 |
| **discover_db** | 是 | 否 | 中 | SQLF_DBTN_DISCOVER | 308 | Uint16 | 第 503 页的『discover_db - 发现数据库』 |
| **dlchktime** | 是 | 否 | 中 | SQLF_DBTN_DLCHKTIME | 9 | Uint32 | 第 504 页的『dlchktime - 检查死锁的时间间隔』 |
| **dyn_query_mgmt** | 否 | 否 | 低 | SQLF_DBTN_DYN_QUERY_MGMT | 604 | Uint16 | 第 504 页的『dyn_query_mgmt - 动态 SQL 和 XQuery 查询管理』 |
| **enable_xmlchar** | 是 | 否 | 无 | SQLF_DBTN_ENABLE_XMLCHAR | 853 | Uint32 | 第 505 页的『enable_xmlchar - 对 XML 启用配置参数转换』 |
| **failarchpath** | 是 | 否 | 无 | SQLF_DBTN_FAILARCHPATH | 826 | char(243) | 第 505 页的『failarchpath - 故障转移日志归档路径』 |
| **hadr_local_host** | 否 | 否 | 无 | SQLF_DBTN_HADR_LOCAL_HOST | 811 | char(256) | 第 506 页的『hadr_local_host - HADR 本地主机名』 |
| **hadr_local_svc** | 否 | 否 | 无 | SQLF_DBTN_HADR_LOCAL_SVC | 812 | char(41) | 第 507 页的『hadr_local_svc - HADR 本地服务名称』 |
| **hadr_peer_window** | 否 | 否 | 低（请参阅注 4） | SQLF_DBTN_HADR_PEER_WINDOW | 914 | Uint32 | 第 507 页的『hadr_peer_window - HADR 对等窗口配置参数』 |
| **hadr_remote_host** | 否 | 否 | 无 | SQLF_DBTN_HADR_REMOTE_HOST | 813 | char(256) | 第 508 页的『hadr_remote_host - HADR 远程主机名』 |
| **hadr_remote_inst** | 否 | 否 | 无 | SQLF_DBTN_HADR_REMOTE_INST | 815 | char(9) | 第 508 页的『hadr_remote_inst - 远程服务器的 HADR 实例名』 |
| **hadr_remote_svc** | 否 | 否 | 无 | SQLF_DBTN_HADR_REMOTE_SVC | 814 | char(41) | 第 509 页的『hadr_remote_svc - HADR 远程服务名称』 |
| **hadr_syncmode** | 否 | 否 | 无 | SQLF_DBTN_HADR_SYNCMODE | 817 | Uint32 | 第 509 页的『hadr_syncmode - 处于对等状态的日志写操作的 HADR 同步方式』 |
| **hadr_timeout** | 否 | 否 | 无 | SQLF_DBTN_HADR_TIMEOUT | 816 | Uint32 | 第 510 页的『hadr_timeout - HADR 超时值』 |
| **indexrec**[2] | 是 | 否 | 中 | SQLF_DBTN_INDEXREC | 30 | Uint16 | 第 449 页的『indexrec - 索引重新创建时间』 |
| **locklist** | 是 | 是 | 高（当它影响升级时） | SQLF_DBTN_LOCK_LIST | 704 | Uint64 | 第 512 页的『locklist -"锁定列表的最大存储量"』 |
| **locktimeout** | 否 | 否 | 中 | SQLF_DBTN_LOCKTIMEOUT | 34 | Sint16 | 第 515 页的『locktimeout - 锁定超时』 |

*表 67. 可配置的数据库配置参数 (续)*

| 参数 | 可联机配置 | 自动 | 性能影响 | 标记 | 标记值 | 数据类型 | 其他信息 |
|---|---|---|---|---|---|---|---|
| **logarchmeth1** | 是 | 否 | 无 | SQLF_DBTN_ LOGARCHMETH1 | 822 | char(252) | 第 516 页的『logarchmeth1 - 主日志归档方法』 |
| **logarchmeth2** | 是 | 否 | 无 | SQLF_DBTN_ LOGARCHMETH2 | 823 | char(252) | 第 517 页的『logarchmeth2 - 辅助日志归档方法』 |
| **logarchopt1** | 是 | 否 | 无 | SQLF_DBTN_LOGARCHOPT1 | 824 | char(243) | 第 518 页的『logarchopt1 - 主日志归档选项』 |
| **logarchopt2** | 是 | 否 | 无 | SQLF_DBTN_LOGARCHOPT2 | 825 | char(243) | 第 518 页的『logarchopt2 - 辅助日志归档选项』 |
| **logbufsz** | 否 | 否 | 高 | SQLF_DBTN_LOGBUFSZ | 33 | Uint16 | 第 519 页的『logbufsz - 日志缓冲区大小』 |
| **logfilsiz** | 否 | 否 | 中 | SQLF_DBTN_LOGFIL_SIZ | 92 | Uint32 | 第 520 页的『logfilsiz - 日志文件大小』 |
| **logindexbuild** | 是 | 是 | 无 | SQLF_DBTN_ LOGINDEXBUILD | 818 | Uint32 | 第 521 页的『logindexbuild - 创建的日志索引页数』 |
| **logprimary** | 否 | 否 | 中 | SQLF_DBTN_LOGPRIMARY | 16 | Uint16 | 第 522 页的『logprimary - 主日志文件数』 |
| **logretain**[3] | 否 | 否 | 低 | SQLF_DBTN_LOG_RETAIN | 23 | Uint16 | 第 523 页的『logretain -"启用日志保留"』 |
| **logsecond** | 是 | 否 | 中 | SQLF_DBTN_LOGSECOND | 17 | Uint16 | 第 523 页的『logsecond - 辅助日志文件数』 |
| **max_log** | 是 | 是 | | SQLF_DBTN_MAX_LOG | 807 | Uint16 | 第 524 页的『max_log - 每个事务的最大日志』 |
| **maxappls** | 是 | 是 | 中 | SQLF_DBTN_MAXAPPLS | 6 | Uint16 | 第 525 页的『maxappls - 最大活动应用程序数』 |
| **maxfilop** | 是 | 否 | 中 | SQLF_DBTN_MAXFILOP | 3 | Uint16 | 第 526 页的『maxfilop - 每个应用程序打开的最大数据库文件数』 |
| **maxlocks** | 是 | 是 | 高（当它影响升级时） | SQLF_DBTN_MAXLOCKS | 15 | Uint16 | 第 526 页的『maxlocks -"锁定升级前锁定列表的最大百分比"』 |
| **min_dec_div_3** | 否 | 否 | 高 | SQLF_DBTN_MIN_DEC_DIV_3 | 605 | Sint32 | 第 528 页的『min_dec_div_3 - 十进制除法，小数位为 3 的』 |
| **mincommit** | 是 | 否 | 高 | SQLF_DBTN_MINCOMMIT | 32 | Uint16 | 第 529 页的『mincommit - 要分组的落实数』 |
| **mirrorlogpath** | 否 | 否 | 低 | SQLF_DBTN_ MIRRORLOGPATH | 806 | char(242) | 第 530 页的『mirrorlogpath - 镜像日志路径』 |
| **newlogpath** | 否 | 否 | 低 | SQLF_DBTN_NEWLOGPATH | 20 | char(242) | 第 531 页的『newlogpath - 更改数据库日志路径』 |
| **num_db_backups** | 是 | 否 | 无 | SQLF_DBTN_NUM_DB_ BACKUPS | 601 | Uint16 | 第 532 页的『num_db_backups - 数据库备份数』 |
| **num_freqvalues** | 是 | 否 | 低 | SQLF_DBTN_NUM_ FREQVALUES | 36 | Uint16 | 第 533 页的『num_freqvalues - 保留的高频值数目』 |
| **num_iocleaners** | 否 | 是 | 高 | SQLF_DBTN_NUM_ IOCLEANERS | 37 | Uint16 | 第 534 页的『num_iocleaners - 异步页清除程序的数目』 |
| **num_ioservers** | 否 | 是 | 高 | SQLF_DBTN_NUM_ IOSERVERS | 39 | Uint16 | 第 535 页的『num_ioservers -"I/O 服务器数"』 |
| **num_log_span** | 是 | 是 | | SQLF_DBTN_NUM_LOG_ SPAN | 808 | Uint16 | 第 536 页的『num_log_span - 跨越的日志数』 |
| **num_quantiles** | 是 | 否 | 低 | SQLF_DBTN_NUM_ QUANTILES | 48 | Uint16 | 第 536 页的『num_quantiles - 列的分位数』 |
| **numarchretry** | 是 | 否 | 无 | SQLF_DBTN_ NUMARCHRETRY | 827 | Uint16 | 第 537 页的『numarchretry - 出错时重试次数』 |
| **overflowlogpath** | 否 | 否 | 中 | SQLF_DBTN_ OVERFLOWLOGPATH | 805 | char(242) | 第 538 页的『overflowlogpath - 溢出日志路径』 |

*表 67. 可配置的数据库配置参数 (续)*

| 参数 | 可联机配置 | 自动 | 性能影响 | 标记 | 标记值 | 数据类型 | 其他信息 |
|---|---|---|---|---|---|---|---|
| **pckcachesz** | 是 | 是 | 高 | SQLF_DBTN_PCKCACHE_SZ | 505 | Sint32 | 第 539 页的『pckcachesz - 程序包高速缓存大小』 |
| **rec_his_retentn** | 否 | 否 | 无 | SQLF_DBTN_REC_HIS_ RETENTN | 43 | Sint16 | 第 541 页的『rec_his_retentn - 恢复历史记录保留期』 |
| **self_tuning_mem** | 是 | 否 | 高 | SQLF_DBTN_SELF_TUNING_ MEM | 848 | Uint16 | 第 543 页的『self_tuning_mem - 自调整内存』 |
| **seqdetect** | 是 | 否 | 高 | SQLF_DBTN_SEQDETECT | 41 | Uint16 | 第 544 页的『seqdetect - 顺序检测标志』 |
| **sheapthres_shr** | 是 | 是 | 高 | SQLF_DBTN_SHEAPTHRES_ SHR | 802 | Uint32 | 第 545 页的『sheapthres_shr - 共享排序的排序堆阈值』 |
| **softmax** | 否 | 否 | 中 | SQLF_DBTN_SOFTMAX | 5 | Uint16 | 第 546 页的『softmax - 恢复范围和软检查点时间间隔』 |
| **sortheap** | 是 | 是 | 高 | SQLF_DBTN_SORT_HEAP | 52 | Uint32 | 第 547 页的『sortheap - 排序堆大小』 |
| **stat_heap_sz** | 是 | 是 | 低 | SQLF_DBTN_STAT_HEAP_SZ | 45 | Uint32 | 第 548 页的『stat_heap_sz -“统计信息堆大小”』 |
| **stmtheap** | 是 | 是 | 中 | SQLF_DBTN_STMT_HEAP | 821 | Uint32 | 第 549 页的『stmtheap -“语句堆大小”』 |
| **trackmod** | 否 | 否 | 低 | SQLF_DBTN_TRACKMOD | 703 | Uint16 | 第 550 页的『trackmod - 启用跟踪已修改页』 |
| **tsm_mgmtclass** | 是 | 否 | 无 | SQLF_DBTN_TSM_ MGMTCLASS | 307 | char(30) | 第 550 页的『tsm_mgmtclass - Tivoli Storage Manager 管理类』 |
| **tsm_nodename** | 是 | 否 | 无 | SQLF_DBTN_TSM_ NODENAME | 306 | char(64) | 第 551 页的『tsm_nodename - Tivoli Storage Manager 节点名』 |
| **tsm_owner** | 是 | 否 | 无 | SQLF_DBTN_TSM_OWNER | 305 | char(64) | 第 551 页的『tsm_owner - Tivoli Storage Manager 所有者名称』 |
| **tsm_password** | 是 | 否 | 无 | SQLF_DBTN_TSM_PASSWORD | 501 | char(64) | 第 552 页的『tsm_password - Tivoli Storage Manager 密码』 |
| **userexit** | 否 | 否 | 低 | SQLF_DBTN_USER_EXIT | 24 | Uint16 | 第 552 页的『userexit -“启用用户出口”』 |
| **util_heap_sz** | 是 | 否 | 低 | SQLF_DBTN_UTIL_HEAP_SZ | 55 | Uint32 | 第 553 页的『util_heap_sz - 实用程序堆大小』 |
| **vendoropt** | 是 | 否 | 无 | SQLF_DBTN_VENDOROPT | 829 | char(242) | 第 553 页的『vendoropt - 供应商选项』< |
| **wlm_collect_int** | 是 | 否 | 低 | SQLF_DBTN_WLM_COLLECT_ INT | 907 | Sint32 | 第 554 页的『wlm_collect_int - 工作负载管理收集时间间隔配置参数』 |

*表 67. 可配置的数据库配置参数 (续)*

| 参数 | 可联机配置 | 自动 | 性能影响 | 标记 | 标记值 | 数据类型 | 其他信息 |
|---|---|---|---|---|---|---|---|
| **注**: SQLF_DBTN_AUTONOMIC_SWITCHES 的位指示许多自动维护配置参数的缺省设置。组成此组合参数的各个位为:<br>1.<br>`Default => Bit 1 on (xxxx xxxx xxxx xxx1): auto_maint`<br>`Bit 2 off (xxxx xxxx xxxx xx0x): auto_db_backup`<br>`Bit 3 on  (xxxx xxxx xxxx x1xx): auto_tbl_maint`<br>`Bit 4 on  (xxxx xxxx xxxx 1xxx): auto_runstats`<br>`Bit 5 off (xxxx xxxx xxx0 xxxx): auto_stats_prof`<br>`Bit 6 off (xxxx xxxx xx0x xxxx): auto_prof_upd`<br>`Bit 7 off (xxxx xxxx x0xx xxxx): auto_reorg`<br>`Bit 8 off (xxxx xxxx 0xxx xxxx): auto_storage`<br>`Bit 9 off (xxxx xxx0 xxxx xxxx): auto_stmt_stats`<br>`0 0 0 D`<br>`Maximum => Bit 1 on (xxxx xxxx xxxx xxx1): auto_maint`<br>`Bit 2 off (xxxx xxxx xxxx xx1x): auto_db_backup`<br>`Bit 3 on  (xxxx xxxx xxxx x1xx): auto_tbl_maint`<br>`Bit 4 on  (xxxx xxxx xxxx 1xxx): auto_runstats`<br>`Bit 5 off (xxxx xxxx xxx1 xxxx): auto_stats_prof`<br>`Bit 6 off (xxxx xxxx xx1x xxxx): auto_prof_upd`<br>`Bit 7 off (xxxx xxxx x1xx xxxx): auto_reorg`<br>`Bit 8 off (xxxx xxxx 1xxx xxxx): auto_storage`<br>`Bit 9 off (xxxx xxx1 xxxx xxxx): auto_stmt_stats`<br>`0 1 F F`<br>2. 有效值（在 sqlutil.h 中定义）为:<br>`SQLF_INX_REC_SYSTEM (0)`<br>`SQLF_INX_REC_REFERENCE (1)`<br>`SQLF_INX_REC_RESTART (2)`<br>3. 有效值（在 sqlutil.h 中定义）为:<br>`SQLF_LOGRETAIN_NO (0)`<br>`SQLF_LOGRETAIN_RECOVERY (1)`<br>`SQLF_LOGRETAIN_CAPTURE (2)`<br>4. 如果将 **hadr_peer_window** 参数设置为一个非零时间值，那么当主数据库处于断开连接对等状态时主数据库可能看起来挂起事务，因为它正在等待来自备用数据库的确认，尽管它未连接至备用数据库。 | | | | | | | |

*表 68. 参考数据库配置参数*

| 参数 | 标记 | 标记值 | 数据类型 | 其他信息 |
|---|---|---|---|---|
| **backup_pending** | SQLF_DBTN_BACKUP_PENDING | 112 | Uint16 | 第 488 页的『backup_pending - 备份暂挂指示器』 |
| **codepage** | SQLF_DBTN_CODEPAGE | 101 | Uint16 | 第 491 页的『codepage - 用于数据库的代码页』 |
| **codeset** | SQLF_DBTN_CODESET | 120 | char(9)[1] | 第 491 页的『codeset - 用于数据库的代码集』 |
| **collate_info** | SQLF_DBTN_COLLATE_INFO | 44 | char(260) | 第 492 页的『collate_info - 整理信息』 |
| **country/region** | SQLF_DBTN_COUNTRY | 100 | Uint16 | 第 492 页的『country/region - 数据库地域代码』 |
| **database_consistent** | SQLF_DBTN_CONSISTENT | 111 | Uint16 | 第 493 页的『database_consistent - 数据库处于一致状态』 |
| **database_level** | SQLF_DBTN_DATABASE_LEVEL | 124 | Uint16 | 第 493 页的『database_level - 数据库发行版级别』 |
| **hadr_db_role** | SQLF_DBTN_HADR_DB_ROLE | 810 | Uint32 | 第 506 页的『hadr_db_role - HADR 数据库角色』 |
| **log_retain_status** | SQLF_DBTN_LOG_RETAIN_STATUS | 114 | Uint16 | 第 516 页的『log_retain_status - 日志保留状态指示器』 |
| **loghead** | SQLF_DBTN_LOGHEAD | 105 | char(12) | 第 521 页的『loghead - 第一个活动日志文件』 |
| **logpath** | SQLF_DBTN_LOGPATH | 103 | char(242) | 第 521 页的『logpath - 日志文件的位置』 |
| **multipage_alloc** | SQLF_DBTN_MULTIPAGE_ALLOC | 506 | Uint16 | 第 531 页的『multipage_alloc - 启用多页文件分配』 |

*表 68. 参考数据库配置参数 (续)*

| 参数 | 标记 | 标记值 | 数据类型 | 其他信息 |
|---|---|---|---|---|
| **numsegs** | SQLF_DBTN_NUMSEGS | 122 | Uint16 | 第 537 页的『numsegs - 缺省 SMS 容器数』 |
| **pagesize** | SQLF_DBTN_PAGESIZE | 846 | Uint32 | 第 539 页的『pagesize - 数据库缺省页大小』 |
| **release** | SQLF_DBTN_RELEASE | 102 | Uint16 | 第 466 页的『release - 配置文件发行版级别』 |
| **restore_pending** | SQLF_DBTN_RESTORE_PENDING | 503 | Uint16 | 第 542 页的『restore_pending - 复原暂挂』 |
| **restrict_access** | SQLF_DBTN_RESTRICT_ACCESS | 852 | Sint32 | 第 542 页的『restrict_access - 数据库具有受限访问配置参数』 |
| **rollfwd_pending** | SQLF_DBTN_ROLLFWD_PENDING | 113 | Uint16 | 第 542 页的『rollfwd_pending - 前滚暂挂指示器』 |
| **territory** | SQLF_DBTN_TERRITORY | 121 | char(5)[2] | 第 550 页的『territory - 数据库地域』 |
| **user_exit_status** | SQLF_DBTN_USER_EXIT_STATUS | 115 | Uint16 | 第 552 页的『user_exit_status - 用户出口状态指示器』 |

**注:**

1. 在 HP-UX、Linux 和 Solaris Operating Environment 上为 char(17)。
2. 在 HP-UX、Linux 和 Solaris 上为 char(33)。

## DB2 管理服务器 (DAS) 配置参数摘要

*表 69. DAS 配置参数*

| 参数 | 参数类型 | 其他信息 |
|---|---|---|
| **认证** | 可配置 | 第 555 页的『authentication - 认证类型 DAS』 |
| **contact_host** | 可联机配置 | 第 555 页的『contact_host - 联系人列表的位置』 |
| **das_codepage** | 可联机配置 | 第 556 页的『das_codepage - DAS 代码页』 |
| **das_territory** | 可联机配置 | 第 556 页的『das_territory - DAS 地域』 |
| **dasadm_group** | 可配置 | 第 556 页的『dasadm_group - DAS 管理权限组名』 |
| **db2system** | 可联机配置 | 第 557 页的『db2system - DB2 服务器系统的名称』 |
| **discover** | 可联机配置 | 第 558 页的『discover - DAS 发现方式』 |
| **exec_exp_task** | 可配置 | 第 558 页的『exec_exp_task - 执行已到期任务』 |
| **jdk_64_path** | 可联机配置 | 第 512 页的『jdk_64_path - 64 位 Java 软件开发者工具箱安装路径 DAS』 |
| **jdk_path** | 可联机配置 | 第 559 页的『jdk_path - Java 软件开发者工具箱安装路径 DAS』 |
| **sched_enable** | 可配置 | 第 559 页的『sched_enable - 调度程序方式』 |
| **sched_userid** | 参考 | 第 560 页的『sched_userid - 调度程序用户标识』 |
| **smtp_server** | 可联机配置 | 第 560 页的『smtp_server - SMTP 服务器』 |
| **toolscat_db** | 可配置 | 第 560 页的『toolscat_db - 工具目录数据库』 |
| **toolscat_inst** | 可配置 | 第 561 页的『toolscat_inst - 工具目录数据库实例』 |
| **toolscat_schema** | 可配置 | 第 561 页的『toolscat_schema - 工具目录数据库模式』 |

## 配置参数节标题

对每个配置参数的描述中都包含下面的某些或所有节标题。在某些情况下，它们是互斥的；例如，如果指定了 [range]，就不需要提供有效值。大多数情况下，这些标题都不需要加以说明。

*表 70. 配置参数节标题的描述*

| 节标题 | 描述和可能的值 |
| --- | --- |
| 配置类型 | 可能的值包括:<br>• 数据库管理器<br>• 数据库<br>• DB2 管理服务器 |
| 适用于 | 如果适用，将列示配置参数适用于的数据服务器类型。可能的值包括:<br>• 客户机<br>• 带有本地和远程客户机的数据库服务器<br>• 带有本地客户机的数据库服务器<br>• DB2 管理服务器<br>• OLAP 函数<br>• 带有本地和远程客户机的分区数据库服务器<br>• 启用联合时带有本地和远程客户机的分区数据库服务器。<br>• 带有本地客户机的卫星数据库服务器 |
| 参数类型 | 可能的值包括:<br>• 可配置（必须重新启动数据库管理器才能使更改生效）<br>• 可联机配置（可联机动态更新，而不需要重新启动数据库管理器）<br>• 供参考（值仅供参考，不能更新） |
| 缺省值 [范围] | 如果适用，那么将列示缺省值和可能的范围，包括 NULL 值或 AUTOMATIC 设置。如果不同平台的范围不同，那么将按平台或平台类型（例如，32 位平台或 64 位平台）来列示值。注意，大多数情况下，未将缺省值列示为范围的一部分。 |
| 计量单位 | 如果适用，那么将列示计量单位。可能的值包括:<br>• 字节<br>• 计数器<br>• 每秒兆字节<br>• 毫秒<br>• 分钟<br>• 页 (4 KB)<br>• 百分比<br>• 秒 |
| 有效值 | 如果适用，那么将列示有效值。此标题与缺省值 [范围] 标题互斥。 |
| 示例 | 如果适用，那么将列示示例。 |
| 传播类 | 如果适用，那么可能的值包括:<br>• 立即<br>• 语句边界<br>• 事务边界<br>• 连接 |
| 分配时间 | 如果适用的话，它指示数据库管理器分配配置参数的时间。 |
| 释放时间 | 如果适用的话，它指示数据库管理器释放配置参数的时间。 |
| 限制 | 如果适用的话，它列示适用于配置参数的所有限制。 |
| 局限性 | 如果适用的话，它列示适用于配置参数的所有局限性。 |
| 建议 | 如果适用的话，它列示适用于配置参数的所有建议。 |
| 使用说明 | 如果适用的话，它列示适用于配置参数的所有使用说明。 |

## 影响代理程序数的配置参数

有许多数据库管理器配置参数与数据库代理程序以及这些代理程序的管理方式相关。

下列数据库管理器配置参数确定创建多少个数据库代理程序以及如何管理它们:

- 代理程序池大小 (*num_poolagents*): 系统中可用的、且要合用的空闲代理程序的总数。此参数的缺省值为 100 和 AUTOMATIC。
- 池中的初始代理程序数 (*num_initagents*): 当启动数据库管理器时，根据此值创建一个工作程序代理程序池。这提高了初始查询的性能。工作程序代理程序全都以空闲代理程序开始。
- 最大连接数 (*max_connections*): 指定每个数据库分区上允许的与数据库管理器系统的最大连接数。
- 最大协调代理程序数 (*max_coordagents*): 当启用了**连接集中器**时，对于分区数据库环境和启用了分区内并行性的环境，此值限制协调代理程序的数目。

## 影响查询优化的配置参数

有一些配置参数影响 SQL 或 XQuery 编译器选择的访问方案。其中的许多参数都适合单一分区数据库环境，而某些参数仅适合分区数据库环境。在同类（硬件相同）的分区数据库环境中，用于每个参数的值应该在所有数据库分区上是相同的。

**注**: 当动态更改配置参数时，优化器可能会由于程序包高速缓存中的旧访问方案而不立即读取已更改的参数值。要复位程序包高速缓存，执行 FLUSH PACKAGE CACHE 命令。

在联合系统中，如果大多数查询都将访问昵称，那么在更改环境之前评估您发送的查询类型。例如，在联合数据库中，缓冲池不高速缓存数据源中的页，这些数据源是 DBMS 和联合系统中的数据。因此，增大缓冲区的大小并不保证当优化器为包含昵称的查询选择访问方案时将考虑其他访问方案备用。但是，优化器可以决定数据源表的本地具体化是否是成本最低的方法，或者是否是排序操作的必需步骤。在这种情况下，增加可用的资源可能会提高性能。

下列配置参数或因子影响 SQL 或 XQuery 编译器选择的访问方案:

- 当创建或改变缓冲池时指定的缓冲池大小

  当优化器选择访问方案时，优化器要考虑将页从磁盘访存至缓冲池的 I/O 成本并估计满足查询所需要的 I/O 次数。估计包括预测缓冲池使用情况，因为不需要其他的物理 I/O 来读取已在缓冲池中的页中的行。

  优化器考虑 SYSCAT.BUFFERPOOLS 系统目录表以及在分区数据库环境中的 SYSCAT.BUFFERPOOLDBPARTITIONS 系统目录表中的 *npages* 列的值。

  读取表的 I/O 成本可影响:

  – 如何连接两个表
  – 是否将使用非集群索引来读取数据
- 缺省度 (dft_degree)

  `dft_degree` 配置参数通过为 CURRENT DEGREE 专用寄存器和 DEGREE 绑定选项提供缺省值来指定并行性。值 1 表示无分区内并行性。值 -1 表示优化器根据处理器数目和查询类型来确定分区内并行度。

  **注**: 除非通过设置 *intra_parallel* 数据库管理器配置参数来启用内部并行处理，否则不会进行内部并行处理。
- 缺省查询优化类 (ft_queryopt)

虽然在编译 SQL 或 XQuery 查询时可以指定查询优化类，但还可以设置缺省查询优化类。

- 活动应用程序平均数 (avg_appls)

  优化器使用 *avg_appls* 参数来帮助估计运行时期间可用于所选访问方案的缓冲池的个数。较高的此参数值会影响优化器，使它选择在缓冲池的使用情况方面更节省的访问方案。如果指定值 1，那么优化器认为整个缓冲池将可用于应用程序。

- 排序堆大小 (sortheap)

  如果要排序的行占用的空间超过排序堆中可用的空间，那么执行几遍排序，每一遍都要对整个行集的某个子集排序。每遍排序的结果存储在缓冲池中的一个系统临时表内，可以将该表写入磁盘。当所有排序都完成时，将这些已排序的子集合并到单个排序的行集。如果排序不需要系统临时表来存储最终的已排序的数据列表，那么可认为该排序是“管道”排序。即，可以按单一的顺序访问方式来读取排序的结果。管道排序的性能优于非管道排序，所以应尽可能使用管道排序。

  当选择访问方案时，优化器要通过下列操作来估计排序操作的成本，包括评估是否可使用管道传送排序:

  – 估计要排序的数据量
  – 查看 *sortheap* 参数，以确定是否有足够的空间通过管道传送排序。

- 锁定列表的最大存储器 (locklist) 和升级前锁定列表的最大百分比 (maxlocks)

  当隔离级别是**可重复读 (RR)**时，优化器考虑 *locklist* 和 *maxlocks* 参数的值，以确定是否可以将行级别锁定升级到表级别锁定。如果优化器估计将对表访问发生锁定升级，那么它为访问方案选择一个表级别锁定，这样就不会在查询执行期间产生锁定升级的开销。

- CPU 速度 (cpuspeed)

  优化器使用 CPU 速度来估计执行特定操作的成本。CPU 成本估计和各种 I/O 成本估计有助于选择查询的最佳访问方案。

  一台机器的 CPU 速度可显著影响所选的访问方案。当安装或迁移数据库时，会自动将此配置参数设置为一个适当的值。除非您要在测试系统上为生产环境建立模型或评估硬件更改的影响，否则不应调整此参数。使用此参数为不同的硬件环境建立模型可以使您找出可为该环境选择的访问方案。要让数据库管理器重新计算此自动配置参数的值，将它设置为 -1。

- 语句堆大小 (stmtheap)

  尽管语句堆的大小不影响优化器选择不同的访问路径，但是，它会影响对复杂的 SQL 或 XQuery 语句执行的优化量。

  如果未将 *stmtheap* 参数设置得足够大，您可能接收到警告，指示无足够可用内存来处理语句。例如，SQLCODE +437(SQLSTATE 01602) 可能指示用于编译语句的优化量小于您请求的量。

- 通信带宽 (comm_bandwidth)

  优化器使用通信带宽来确定访问路径。优化器使用此参数中的值来估计在分区数据库环境的各数据库分区服务器之间执行特定操作的成本。

- 应用程序堆大小 (applheapsz)

大模式要求应用程序堆中有足够的空间。

## 配置 max_coordagents 和 max_connections 时的限制和行为

在版本 9.5 中，*max_coordagents* 和 *max_connections* 参数的缺省值将为 AUTOMATIC，并且 *max_coordagents* 设置为 200 且 *max_connections* 设置为 -1（也就是说，设置为 *max_coordagents* 的值）。这些设置将集中器设置为 OFF。

联机配置 *max_coordagents* 或 *max_connections* 时，您需要了解一些限制和行为:

- 如果 *max_coordagents* 的值增大，那么设置将立即生效并且允许创建新的协调代理程序的新请求。如果该值减小，那么协调代理程序数将不会立即减小。确切地说，协调代理程序数将不再增大，并且现有协调代理程序在完成它们的当前工作集后可能终止，以便减小协调代理程序总数。将不处理需要协调代理程序的新工作请求，直到协调代理程序总数小于新值并且一个协调代理程序变得可用为止。
- 如果 *max_connections* 的值增大，那么设置将立即生效并且允许先前由于此参数而被阻塞的新连接。如果该值减小，那么数据库管理器将不会主动终止现有连接；相反，将不允许新的连接，直到终止了足够多的现有连接以使值减小到小于新的最大值为止。
- 如果 *max_connections* 设置为 -1（缺省值），那么允许的最大连接数与 *max_coordagents* 相同；当以脱机或联机方式更新 *max_coordagents* 时，也会更新允许的最大连接数。

以联机方式更改 *max_coordagents* 或 *max_connections* 的值时，您不能更改它以便连接集中器打开（如果它关闭）或关闭（如果它打开）。例如，如果在 DB2START 时 *max_coordagents* 小于 *max_connections*（集中器为打开状态），那么对这两个参数执行的所有联机更新必须保持关系 *max_coordagents* < *max_connections*。同样，如果在 DB2START 时 *max_coordagents* 大于或等于 *max_connections*（集中器为关闭状态），那么执行的所有联机更新必须保持此关系。

当您执行此类联机更新时，数据库管理器的更新操作不会失败，它而是会延迟更新。将返回警告 SQL1362W 消息，这类似于更新指定了 IMMEDIATE 的数据库管理器配置参数的任何情况，但不可能出现这种情况。

将 *max_coordagents* 或 *max_connections* 设置为 AUTOMATIC 时，应出现下列行为:

- 可以使用一个起始值和 AUTOMATIC 设置来配置这两个参数。例如，以下命令使值 200 和 AUTOMATIC 与 *max_coordagents* 参数关联:

  ```
  UPDATE DBM CONFIG USING max_coordagents 200 AUTOMATIC
  ```

  这些参数始终有一个值与它们相关，要么是设置为缺省值的值，要么是指定的某个值。如果在更新任一参数时仅指定了 AUTOMATIC（也就是说，未指定值），并且该参数先前有一个值与它关联，那么该值将保留。仅 AUTOMATIC 设置受影响。

  **注:** 当集中器打开时，即使这两个配置参数设置为 AUTOMATIC，指定给它们的值也很重要。
- 如果两个参数都设置为 AUTOMATIC，那么数据库管理器允许连接数和协调代理程序数根据需要增大以适合工作负载。但是，下列警告适用:
  1. 当集中器关闭时，数据库管理器保持 1: 1 的比率: 对于每个连接，只有一个协调代理程序。

2. 当集中器打开时，数据库管理器尝试保持参数中的值设置的协调代理程序数与连接数的比率。

   **注:**

   – 用于保持比率的方法被设计为非侵入的，并且不能保证精确地保持比率。在此方案中，始终允许新的连接，虽然这些连接可能必须等待可用的协调代理程序。将根据需要创建新的协调代理程序以保持比率。当连接终止时，数据库管理器还可能终止协调代理程序来保持比率。

   – 数据库管理器将不会减小您设置的比率。将设置的 *max_coordagents* 和 *max_connections* 的初始值视为下限。

- 可以通过各种方法（例如，CLP 或 API）显示这两个参数的当前值和延迟的值。显示的值始终是用户设置的值。例如，如果发出了以下命令，然后启动 30 个在实例上执行工作的并发连接，那么对 *max_connections* 和 *max_coordagents* 显示的值仍为 20 和 AUTOMATIC:

  ```
  UPDATE DBM CFG USING max_connections 20 AUTOMATIC,
    max_coordagents 20 AUTOMATIC
  ```

  要确定当前正在运行监视元素的连接和协调代理程序的实际数目，还可以使用运行状况监视器。

- 如果 *max_connections* 设置为 AUTOMATIC 并且值大于 *max_coordagents*（以便集中器打开），并且 *max_coordagents* 未设置为 AUTOMATIC，那么数据库管理器将允许无数个仅使用有限数目的协调代理程序的连接。

  **注:** 连接可能必须等待可用的协调代理程序。

对 *max_coordagents* 和 max_connections 配置参数使用 AUTOMATIC 选项仅在下列两种情况下有效:

1. 这两个参数都设置为 AUTOMATIC
2. 集中器已启用并且 *max_connections* 设置 AUTOMATIC，而 *max_coordagents* 未设置为 AUTOMATIC。

将 AUTOMATIC 用于这些参数的所有其他配置都将被阻塞并且返回 SQL6112N，其中的原因码说明这两个参数的 AUTOMATIC 的有效设置。

# 数据库管理器配置参数

## agent_stack_sz - 代理程序堆栈大小

此参数确定 DB2 为每个代理程序分配的虚拟内存。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

缺省值 [范围]

**Linux（32 位）**
256 [16 – 1024]

**Linux（64 位）和 UNIX**
1024 [256 – 32768]

**Windows**
16 [8 – 1000]

计量单位
页 (4 KB)

分配时间
当初始化代理程序为应用程序工作时

释放时间
当代理程序完成应用程序要做的工作时

对于给定的一组应用程序，可使用此参数优化服务器的内存使用率。与用于简单查询的空间相比，越复杂的查询将使用越多的堆栈空间。

此参数用来设置在 Windows 环境中的每个代理程序的初始落实堆栈大小。缺省情况下，每个代理程序堆栈可增长至缺省保留堆栈大小 256 KB（64 个 4 KB 页）。此限制对于大多数数据库操作已足够。在 UNIX 和 Linux 上，*agent_stack_sz* 将四舍五入为下一个更大的基于 2 的幂的值。UNIX 的缺省设置对于大多数工作负载来说应该足够了。

但是，在准备大型 SQL 或 XQuery 语句时，代理程序可能用完堆栈空间，且系统将生成堆栈溢出异常（0xC00000FD）。当发生此情况时，服务器将关闭，因为该错误不可恢复。

**注**: 在版本 9.5 和更高版本中，将返回 sqlcode -973 而不会产生堆栈溢出异常。

可以通过将 *agent_stack_sz* 的值设置为大于 64 页的缺省保留堆栈大小来增大代理程序堆栈大小。注意，当 *agent_stack_sz* 的值大于缺省保留堆栈大小时，Windows 操作系统会将其四舍五入为最接近的 1 MB 的倍数；将代理程序堆栈大小设置为 128 个 4 KB 的页实际上为每个代理程序保留 1 MB 的堆栈。将 *agent_stack_sz* 的值设置为小于缺省保留堆栈大小不会对最大限制产生影响，因为在必要时，堆栈将增长至缺省保留堆栈大小。在此情况下，*agent_stack_sz* 的值是创建代理程序时堆栈的初始落实内存。

可以通过使用 db2hdr 实用程序更改 `db2syscs.exe` 文件的头信息来更改缺省保留堆栈大小。更改缺省保留堆栈大小将影响所有线程，而更改 *agent_stack_sz* 仅影响代理程序的堆栈大小。使用 db2hdr 实用程序更改缺省堆栈大小的优点是提供更佳的详细程度，从而允许以最小的必要堆栈大小来设置堆栈大小。但是，您必须停止并重新启动 DB2 以便使对 `db2syscs.exe` 的更改生效。

**建议**: 如果您要在 32 位环境中使用大量 XML 数据或复杂 XML 数据，那么应将 *agent_stack_sz* 更新为至少 256 个 4 KB 的页。非常复杂的 XML 模式可能要求将 *agent_stack_sz* 设置为尽量接近上限的值，以避免在模式注册或 XML 文档验证过程中出现堆栈溢出异常。

如果您的环境符合下列条件，可减小堆栈大小以使更多地址空间可供其他客户机使用:

- 仅包含简单应用程序（例如，小型的 OLTP），其中从不会有复杂查询
- 需要相当大量的并发客户机（例如，超过 100 个）。

在 Windows 操作系统上，代理程序堆栈大小和并发客户机数是成反比的：堆栈大小越大，那么可运行的并发客户机的可能数量就越少。发生这种情况的原因是 Windows 平台上的地址空间有限。

## agentpri - 代理程序的优先级

在版本 9.5 中不推荐使用此参数，但是版本 9.5 之前的数据服务器和客户机仍然使用此参数。为此配置参数指定的所有值都将完全像在先前版本中一样继续工作，并且此参数将继续完全受支持。如果此参数用于工作负载管理 (WLM)，那么将忽略 WLM 服务类代理程序优先级。

**注:** 下列信息仅适用于版本 9.5 之前的数据服务器和客户机。

此参数通过操作系统调度程序来控制所有代理程序和其他数据库管理器实例进程和线程的优先级。此优先级确定如何将 CPU 时间分配给与在该机器上运行的其他进程和线程相关的数据库管理器进程、代理程序和线程。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

**AIX**

-1 (system)[ 41 - 125 ]

**其他 UNIX**

-1 (system)[ 41 - 128 ]

**Windows**

-1 (system)[ 0 - 6 ]

**Solaris**

-1 (system) [ 0 - 59 ]

将此参数设置为 -1 或 system 时，不会执行任何特殊操作，并且将以操作系统调度所有进程和线程的正常方式来调度数据库管理器。将此参数设置为除 -1 或 system 以外的值时，数据库管理器将在静态优先级设置为此参数值的情况下创建其进程和线程。因此，此参数允许您控制数据库管理器进程和线程（在分区数据库环境中，这还包括协调和子代理程序、并行系统控制器以及 FCM 守护程序）在机器上执行时所用的优先级。

可使用此参数来增大数据库管理器吞吐量。用于设置此参数的值取决于正在运行数据库管理器的操作系统。例如，在 Linux 或 UNIX 环境中，低数值代表高优先级。当将该参数设置为在 41 和 125 之间的一个值时，数据库管理器在该参数的值设置为 UNIX

静态优先级的情况下创建其代理程序。这在 Linux 和 UNIX 环境中很重要，因为低数值代表数据库管理器的高优先级，但其他进程（包括应用程序和用户）可能因为不能获取足够的 CPU 时间而遇到延迟。应该使此参数的设置与机器上其他预期的活动相平衡。

**限制:**

- 如果在 Linux 和 UNIX 平台上将此参数设置为非缺省值，那么不能使用控制器来改变代理程序的优先级。
- 在 Solaris 操作系统上，不应该更改缺省值（-1）。更改该缺省值会将 DB2 进程的优先级设置为实时，这可能会垄断系统上的所有可用资源。

**建议**: 最初应使用缺省值。此值提供了对其他用户/应用程序的响应时间与数据库管理器吞吐量之间关系的很好的折衷办法。

如果关心的是数据库性能，可以使用基准程序技术来确定此参数的最佳设置。提高数据库管理器的优先级时应仔细，因为可能会极大地降低其他用户进程的性能，特别是当 CPU 利用率很高的时候更是如此。提高数据库管理器进程和线程的优先级可显著地提高性能。

## aslheapsz - 应用程序支持层堆大小

应用程序支持层堆表示本地应用程序和其关联的代理程序之间的通信缓冲区。此缓冲区被分配为每个已启动的数据库管理器代理程序所共享的内存。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

15 [1 - 524 288]

**计量单位**

页 (4 KB)

**分配时间**

当为本地应用程序启动数据库管理器代理程序时

**释放时间**

当数据库管理器代理程序终止时

如果对数据库管理器的请求或其相关联的应答不适合该缓冲区，那么该请求和应答将分成两个或更多的发送-接收对。应将此缓冲区的大小设置为可使用单个发送-接收对来处理大多数请求。请求的大小基于保存下列各项所需的存储器:

- 输入 SQLDA
- SQLVAR 中的所有相关数据
- 输出 SQLDA

- 一般不超过 250 个字节的其他字段。

除了此通信缓冲区外，此参数也用于两个其他目的:

- 它用来确定在打开分块游标时的 I/O 块大小。这个用于分块游标的内存是在应用程序专用地址空间之外分配的，所以应确定要分配给每个应用程序的最佳专用内存量。如果数据服务器运行时客户机不能为分块游标分配应用程序的专用内存之外的空间，那么将打开非分块游标。
- 它用来确定代理程序和 db2fmp 进程之间的通信大小。（db2fmp 进程可以是用户定义的函数或受防护的存储过程。）字节的数目是从系统上活动的每个 db2fmp 进程或线程的共享内存分配的。

数据库管理器将从本地应用程序发送的数据接收到从查询堆中分配的一组相邻内存中。*aslheapsz* 参数用于确定查询堆（用于本地和远程客户机）的初始大小。查询堆的最大大小由 *query_heap_sz* 参数定义。

**建议**: 如果您的应用程序的请求通常较小，并且该应用程序在内存受约束的系统上运行，那么您可能希望减小此参数的值。如果查询一般都很大，需要多个发送和接收请求，并且您的系统不受内存约束，那么您可能希望增大此参数的值。

使用如下公式计算 *aslheapsz* 的最小页数:

```
aslheapsz >= ( sizeof(input SQLDA)
                                + sizeof(each input SQLVAR)
                                + sizeof(output SQLDA)
                                + 250 ) / 4096
```

其中 sizeof(x) 是 x 的字节大小，它计算给定输入或输出值的页数。

还应考虑此参数对分块游标的数目和潜在大小的影响。如果所传送的行的数目或大小较大（例如，如果数据量大于 4096 个字节），那么大行块可能获得更佳性能。但是，由于较大的记录块会增大每个连接的工作集内存大小，所以要加以折衷。

大记录块还可能导致比应用程序实际需要的更多的访存请求。可通过在应用程序中的 SELECT 语句上使用 OPTIMIZE FOR 子句来控制访存请求数。

## audit_buf_sz - 审计缓冲区大小

此参数指定当审计数据库时使用的缓冲区的大小。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

0 [0 - 65 000 ]

**计量单位**

页 (4 KB)

**分配时间**

当启动 DB2 时

**释放时间**

当停止 DB2 时

此参数的缺省值是零 (0)。如果该值为零 (0)，那么不使用该审计缓冲区。如果该值大于零 (0)，就会为审计缓冲区分配空间，以用于放置审计设施所生成的审计记录。该值乘以 4KB 页就是为审计缓冲区分配的空间容量。不能动态分配审计缓冲区；即必须停止 DB2，然后重新启动它，此参数的新值才会生效。

通过将此参数从缺省值更改为大于零 (0) 的某个值，该审计设施将记录写入磁盘与执行生成审计记录的语句将异步进行。这使 DB2 性能比保留该参数值为零 (0) 时有所提高。该值为零 (0) 意味着，审计设施将记录写入磁盘与执行生成审计记录的语句将同步进行（同时进行）。审计期间进行同步操作会降低在 DB2 中运行的应用程序的性能。

## authentication - 认证类型

指定并确定如何进行以及在何处进行用户认证。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

SERVER [CLIENT; SERVER; SERVER_ENCRYPT; DATA_ENCRYPT; DATA_ENCRYPT_CMP; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT ]

如果 **authentication** 的值为 SERVER，那么会将用户标识和密码从客户机发送至服务器，以便可在服务器上进行认证。值 SERVER_ENCRYPT 与 SERVER 提供相同的行为，只不过前者将对通过网络发送的任何用户标识和密码进行加密。

如果值为 DATA_ENCRYPT，那么意味着服务器接受已加密的 SERVER 认证方案并对用户数据进行加密。该认证与 SERVER_ENCRYPT 的工作方式完全相同。

使用此认证类型时，加密以下用户数据:

- SQL 语句
- SQL 程序变量数据
- 来自处理 SQL 语句的服务器的输出数据并且包括对数据的描述
- 从查询获得的某些或所有答案集数据
- 大对象（LOB）数据流动
- SQLDA 描述符

如果值为 DATA_ENCRYPT_CMP，那么意味着服务器接受已加密的 SERVER 认证方案并对用户数据进行加密。另外，此认证类型与不支持 DATA_ENCRYPT 认证类型的较早产品相兼容。允许这些产品使用 SERVER_ENCRYPT 认证类型进行连接，并且不对用户数据进行加密。支持新认证类型的产品必须使用该认证类型。此认证类型仅在服务器的数据库管理器配置文件中有效，而在 CATALOG DATABASE 命令上使用该认证类型无效。

**注:** 对于符合标准（在『符合标准』主题中定义）的配置，SERVER 是唯一受支持的值。

如果值为 CLIENT，那么所有认证都将在客户机上进行。不需要在服务器上进行认证。

如果值为 KERBEROS，那么意味着将在 Kerberos 服务器上使用 Kerberos 认证安全性协议来执行认证。如果支持 Kerberos 安全性系统的服务器和客户机上的认证类型为 KRB_SERVER_ENCRYPT，那么有效的系统认证类型是 KERBEROS。如果客户机不支持 Kerberos 安全性系统，那么有效的系统认证类型等价于 SERVER_ENCRYPT。

如果值为 GSSPLUGIN，那么意味着将使用外部的基于 GSSAPI 的安全性机制来执行认证。如果支持 GSSPLUGIN 安全性机制的服务器和客户机上的认证类型为 GSS_SERVER_ENCRYPT，那么有效的系统认证类型为 GSSPLUGIN（即，要求客户机支持服务器的其中一个插件）。如果客户机不支持 GSSPLUGIN 安全性机制，那么有效的系统认证类型等价于 SERVER_ENCRYPT。

**建议:** 通常，对本地客户机使用缺省值（SERVER）就可以了。如果远程客户机正连接至数据库服务器，那么建议您使用 SERVER_ENCRYPT 来保护用户标识和密码。

## catalog_noauth - 没有权限时允许的编目

此参数指定用户是否可以编目和取消编目数据库和节点，或者 DCS 和 ODBC 目录，而不需要 SYSADM 权限。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

**带有本地和远程客户机的数据库服务器**

NO [ NO (0) — YES (1) ]

**客户机；带有本地客户机的数据库服务器**

YES [ NO (0) — YES (1) ]

此参数的缺省值 (0) 指示 SYSADM 权限是必需的。当将此参数设置为 1（是）时，就不需要 SYSADM 权限。

## clnt_krb_plugin - 客户机 Kerberos 插件

此参数指定要用于客户端认证和本地授权的缺省 Kerberos 插件库的名称。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

Null 或 IBMkrb5 [任何有效字符串]

缺省情况下，在 Linux 和 UNIX 系统上，值为 NULL，而在 Windows 操作系统上为 IBMkrb5。当使用 KERBEROS 认证来对客户机进行认证时，或者执行本地认证并且 DBM CFG 中的认证类型为 KERBEROS 时，使用该插件。

## clnt_pw_plugin - 客户机用户标识密码插件

此参数指定要用于客户端认证和本地授权的用户标识密码插件库的名称。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

Null [任何有效字符串]

缺省情况下，值为 NULL 并且使用 DB2 提供的用户标识密码插件库。当使用 CLIENT 认证来对客户机进行认证时，或者执行本地认证并且 DBM CFG 中的认证类型为 CLIENT、SERVER、SERVER_ENCRYPT 或 DATA_ENCRYPT 时，使用该插件。对于非 root 用户安装，如果使用了 DB2 用户标识和密码插件库，那么 db2rfe 命令必须在运行才能使用 DB2 产品。

## cluster_mgr - 集群管理器名称

此参数使数据库管理器能够将增量集群配置更改告知指定的集群管理器。

**配置类型**
数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的多分区数据库服务器

**参数类型**
参考

**缺省值**
无缺省值

**有效值**

- TSA

此参数是在高可用性集群配置期间使用 DB2 高可用性实例配置实用程序 (db2haicu) 设置的。

## comm_bandwidth - 通信带宽

此参数通过指示数据库分区服务器之间的带宽来帮助查询优化器确定访问路径。

**配置类型**
数据库管理器

**适用于** 带有本地和远程客户机的分区数据库服务器

**参数类型**
可联机配置

**传播类** 语句边界

**缺省值 [范围]**
-1 [.1 - 100 000 ]

值 -1 导致将该参数值复位为缺省值。缺省值是根据底层通信适配器的速度计算的。对于使用千兆以太网的系统，可使用值 100。

**计量单位**
每秒兆字节

为通信带宽计算的值以每秒兆字节计，查询优化器使用它来估计在一个分区数据库系统的数据库分区服务器之间执行特定操作的成本。该优化器不为客户机和服务器之间的通信成本建立模型，所以此参数应该只反映数据库分区服务器之间的标称带宽（如果有）。

您可以显式地设置此值，以便在您的测试系统上建立一个生产环境模型，或者评估硬件升级的效果。

**建议**：如果您要建立不同的环境模型，那么只应调整此参数。

优化器使用通信带宽来确定访问路径。在更改此参数之后应考虑重新绑定应用程序（使用 REBIND PACKAGE 命令）。

## conn_elapse - 连接耗用时间

此参数指定在两个数据库分区服务器之间建立 TCP/IP 连接所用的秒数。

**配置类型**
数据库管理器

**适用于** 带有本地和远程客户机的分区数据库服务器

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
10 [0–100]

**计量单位**
秒

如果连接尝试在此参数指定的时间内成功，那么建立了通信。如果失败，那么进行另一次尝试来建立通信。如果尝试连接的次数达到 *max_connretries* 参数指定的次数且始终超时，那么发出错误。

## cpuspeed - CPU 速度

此参数反映安装了数据库的机器的 CPU 速度。

在创建 DB2 实例时，将针对操作系统多次运行一个小工作单元。CPU 速度为完成此工作单元所需的平均时间。

**配置类型**
数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**
可联机配置

**传播类** 语句边界

**缺省值 [范围]**
-1 [ $1x10^{-10}$ - 1 ] 根据度量程序的运行情况，值 -1 将导致该参数值被复位。

**计量单位**
毫秒

如果基准程序结果不可用、在该文件中未找到 IBM RS/6000 型号 530H 的数据或者在该文件中未找到您使用的机器的数据，那么就会执行此程序。

您可以显式地设置此值，以便在您的测试系统上建立一个生产环境模型，或者评估硬件升级的效果。通过将它设置为 -1，以重新计算 *cpuspeed*。

**建议**：如果您要建立不同的环境模型，那么只应调整此参数。

优化器在确定访问路径时使用 CPU 速度。在更改此参数之后应考虑重新绑定应用程序（使用 REBIND PACKAGE 命令）。

## dft_account_str - 缺省对方付费帐户

此参数充当记帐标识的缺省后缀。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

Null [任何有效字符串]

对于每个应用程序连接请求，将一个由 DB2 Connect 生成的前缀和用户提供的后缀组成的记帐标识从应用程序请求器发送至 DRDA 应用程序服务器。此记帐信息给系统管理员提供一种使资源使用与每个用户访问关联的机制。

**注:** 此参数仅适用于 DB2 Connect。

该后缀是通过应用程序调用 `sqlesact()` API 或用户设置环境变量 DB2ACCOUNT 来提供的。如果 API 或环境变量未提供后缀，那么 DB2 Connect 将此参数的值用作缺省后缀值。对于没有能力向 DB2 Connect 转发记帐字符串的较早版本的数据库客户机（版本 2 之前的任何版本），此参数尤其有用。

**建议:** 使用下列项设置此记帐字符串:

- 字母（A 至 Z）
- 数字（0 至 9）
- 下划线（_）。

## dft_monswitches - 缺省数据库系统监视器开关

此参数允许您设置许多开关，这些开关在内部分别由该参数的一位表示。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**注**: 如果在修改 dft_mon_xxxx 开关设置前显式连接至实例，那么更改将立即生效。否则，设置在下次重新启动实例时生效。

**缺省值** 所有开关关闭，但 dft_mon_timestamp 除外，该开关缺省情况下是打开的

该参数是唯一的，因此可以通过设置下列参数来单独地更新这些开关中的每一个开关:

**dft_mon_uow**

快照监视器的工作单元（UOW）开关的缺省值

**dft_mon_stmt**

快照监视器的语句开关的缺省值

**dft_mon_table**

快照监视器的表开关的缺省值

**dft_mon_bufpool**

快照监视器的缓冲池开关的缺省值

**dft_mon_lock**

快照监视器的锁定开关的缺省值

**dft_mon_sort**

快照监视器的排序开关的缺省值

**dft_mon_timestamp**

快照监视器的时间戳记开关的缺省值

**建议**: 任何打开的开关（dft_mon_timestamp 除外）指示数据库管理器收集与该开关相关的监视器数据。收集更多监视器数据将增加数据库管理器的开销，这会影响系统性能。在 CPU 利用率达到 100% 时应将 dft_mon_timestamp 开关关闭。当此情况发生时，发出时间戳记所需的 CPU 时间急剧增长。因此，如果时间戳记开关已关闭，那么在监视器开关控制下的其他数据的总体成本大幅减少。

当应用程序发出其第一个监视请求（例如，设置开关、激活事件监视器、生成快照）时，所有监视应用程序继承这些缺省开关设置。仅当想从启动数据库管理器起开始收集数据时，才应打开配置文件中的开关。（否则，每个监视应用程序可设置其自己的开关，并且该程序收集的数据与设置开关的时间有关。）

## dftdbpath - 缺省数据库路径

此参数包含用来在数据库管理器下创建数据库的缺省文件路径。如果创建数据库时未指定路径，那么在 *dftdbpath* 参数指定的路径下创建该数据库。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器

- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

**UNIX** 实例所有者的主目录 [任何现有路径]

**Windows**

安装了 DB2 的驱动器 [任何现有路径]

在分区数据库环境中，应确保正在其上创建数据库的路径不是安装了 NFS 的路径（在 Linix 和 UNIX 平台上）或网络驱动器（在 Windows 环境中）。指定的路径必须实际存在于每个数据库分区服务器上。为避免混淆，最好指定以本地方式安装在每个数据库分区服务器上的路径。该路径的最大长度为 205 个字符。系统将数据库分区名追加至该路径的末尾。

假如数据库可扩充至更大，并且很多用户可能要创建数据库（根据环境和意向），如果能在指定的位置上创建和存储所有数据库，通常会很方便。这样也能将数据库与其他应用程序和数据分离，保持了数据库完整性并便于备份和恢复。

对于 Linux 和 UNIX 环境，*dftdbpath* 名称长度不能超过 215 个字符且必须为有效的绝对路径名。对于 Windows，*dftdbpath* 可以为盘符，可选择后跟冒号。

**建议**：如果有可能，将高容量数据库放在不同于其他被频繁访问的数据（如操作系统文件和数据库日志）所在的磁盘上。

## diaglevel -“诊断错误捕获级别”

此参数指定将记录在 db2diag.log 文件中的诊断错误类型。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

3 [ 0 — 4 ]

此参数的有效值为:

- **0** - 没有捕获到诊断数据
- **1** - 仅严重错误
- **2** - 所有错误

- **3** - 所有错误和警告
- **4** - 所有错误、警告以及参考消息

*diagpath* 配置参数用来指定一个目录，它将包含根据 *diaglevel* 参数的值可能会生成的错误文件、事件日志文件、警报日志文件以及任何转储文件。

**使用说明**

- **diaglevel** 的动态行为不会扩展到所有进程。
- db2sysc DB2 服务器进程可以检测到动态更改，例如，当您对实例连接发出 UPDATE DATABASE MANAGER CONFIGURATION 命令时。
- DB2 客户机和应用程序启动时，将使用 **diaglevel** 配置参数设置且不检测任何动态更改。
- 要帮助解决问题，您可以增大该参数的值以收集更多问题确定数据。

## diagpath -“诊断数据目录路径”

此参数允许您指定 DB2 诊断信息的标准路径。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

Null [任何有效路径名，`'"$h"'`、`'"`*pathname* `$h"'`、`'"$n"'`、`'"`*pathname* `$n"'`、`'"$h$n"'` 或 `'"`*pathname* `$h$n"'`]

DB2 版本 9.5 修订包 5 和更高版本的修订包中提供了值 `'"$h"'`、`'"`*pathname* `$h"'`、`'"$n"'`、`'"`*pathname* `$n"'`、`'"$h$n"'` 和 `'"`*pathname* `$h$n"'`。

根据您使用的平台不同，此诊断数据目录中可能包含转储文件、陷阱文件、错误日志、通知文件、警报日志文件和“首次出现数据集合”（FODC）程序包。

如果此参数为 Null，那么诊断信息将写入下列其中一个目录或文件夹中的文件:

- 在 Windows 环境中:
  - 用户数据文件（例如，实例目录下的文件）被写入不同于安装了代码的位置，如下所示:
    - 在 Windows Vista 环境中，用户数据文件将被写入 `ProgramData\IBM\DB2\`。
    - 在 Windows 2003 和 XP 环境中，用户数据文件将被写入 `Documents and Settings\All Users\Application Data\IBM\DB2\`*Copy Name*，其中 *Copy Name* 是 DB2 副本的名称。

- 在 Linux 和 UNIX 环境中: 信息将被写入 *INSTHOME*/sqllib/db2dump，其中 *INSTHOME* 是实例的主目录。

要分割诊断数据目录路径以收集每个物理主机的诊断信息，请将此参数设置为下列其中一个值:

- 分割缺省诊断数据目录路径:

  ```
  db2 update dbm cfg using diagpath '"$h"'
  ```

  这将在缺省诊断数据目录下使用主机名创建一个子目录，如下所示:

  *Default_diagpath*/HOST_*hostname*
- 分割为您自己指定的诊断数据目录路径（*pathname* 与 $h 之间有个空格）:

  ```
  db2 update dbm cfg using diagpath '"pathname $h"'
  ```

  这将在为您自己指定的诊断数据目录下使用主机名创建一个子目录，如下所示:

  *pathname*/HOST_*hostname*

要分割诊断数据目录路径以收集每个物理主机的每个数据库分区的诊断信息，请将此参数设置为下列其中一个值:

- 分割缺省诊断数据目录路径:

  ```
  db2 update dbm cfg using diagpath '"$n"'
  ```

  这将在缺省诊断数据目录下使用分区号为每个分区创建一个子目录，如下所示:

  *Default_diagpath*/NODE*number*
- 分割为您自己指定的诊断数据目录路径（*pathname* 与 $n 之间有个空格）:

  ```
  db2 update dbm cfg using diagpath '"pathname $n"'
  ```

  这将在为您自己指定的诊断数据目录下使用分区号为每个分区创建一个子目录，如下所示:

  *pathname*/NODE*number*

要分割诊断数据目录路径以收集每个物理主机以及每个物理主机的每个数据库分区的诊断信息，请将此参数设置为下列其中一个值:

- 分割缺省诊断数据目录路径:

  ```
  db2 update dbm cfg using diagpath '"$h$n"'
  ```

  这将在缺省诊断数据目录下使用主机名和分区号为主机上的每个逻辑分区创建一个子目录，如下所示:

  *Default_diagpath*/HOST_*hostname*/NODE*number*
- 分割为您自己指定的诊断数据目录路径（*pathname* 与 $h$n 之间有个空格）:

  ```
  db2 update dbm cfg using diagpath '"pathname $h$n"'
  ```

  这将在为您自己指定的诊断数据目录下使用主机名和分区号为主机上的每个逻辑分区创建一个子目录，如下所示:

  *pathname*/HOST_*hostname*/NODE*number*

  例如，一个名为 boson 的 AIX 主机有 3 个数据库分区，节点号分别为 0、1 和 2。此目录的列表输出示例与以下内容相似:

```
usr1@boson /home/user1/db2dump->ls -R *
HOST_boson:

HOST_boson:
NODE0000 NODE0001 NODE0002

HOST_boson/NODE0000:
db2diag.log db2eventlog.000 db2resync.log db2sampl_Import.msg events usr1.nfy

HOST_boson/NODE0000/events:
db2optstats.0.log

HOST_boson/NODE0001:
db2diag.log db2eventlog.001 db2resync.log usr1.nfy stmmlog

HOST_boson/NODE0001/stmmlog:
stmm.0.log

HOST_boson/NODE0002:
db2diag.log db2eventlog.002 db2resync.log usr1.nfy
```

### 使用说明

- 为了避免操作系统 Shell 解释某些 Linux 和 UNIX 系统上的 $ 符号，必须在双引号两边添加单引号，如语法中所示。
- 在 CLP 交互方式下，或者从输入文件中读取和执行命令时，不需要添加双引号。
- $h 和 $n 不区分大小写。
- 如果指定了对每个数据库分区进行诊断数据目录路径分割（$n 或 $h$n），那么始终将为每个主机都创建 NODE0000 目录。如果在创建了 NODE0000 目录的主机上不存在数据库分区 0，那么可以忽略 NODE0000 目录。
- 要检查是否成功分割了诊断数据目录路径的设置，请执行以下命令:

  ```
  db2 get dbm cfg | grep DIAGPATH
  ```

  如果成功分割了诊断数据目录路径，那么将返回值 $h、$n 或 $h$n，并且这些值前面有个空格。例如，所返回的输出与以下内容相似:

  ```
  Diagnostic data directory path           (DIAGPATH) = /home/usr1/db2dump/ $h$n
  ```

- **diagpath** 的动态行为不会扩展到所有进程。
- db2sysc DB2 服务器进程可以检测到动态更改，例如，当您对实例连接发出 UPDATE DATABASE MANAGER CONFIGURATION 命令时。
- DB2 客户机和应用程序启动时，将使用 **diagpath** 配置参数设置且不检测任何动态更改。
- 为了提高性能，可以在非分区数据库环境中使用 **diagpath** 配置参数的缺省设置。在分区数据库环境中，可以将主机中的本地存储器用于 **diagpath** 配置参数。

在版本 9.5 中，全局级 **DB2INSTPROF** 的缺省值存储在上面所显示的新位置。指定运行时数据文件的位置的其他概要文件注册表变量应查询 **DB2INSTPROF** 的值。其他变量包括:

- **DB2CLIINIPATH**
- **DIAGPATH**
- **SPM_LOG_PATH**

## dir_cache - 目录高速缓存支持

此参数确定是否将数据库、节点和 DCS 目录文件高速缓存到内存中。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

Yes [ Yes; No ]

**分配时间**

- 当应用程序发出它的第一个连接时，分配应用程序目录高速缓存
- 当启动（db2start）数据库管理器实例时，分配服务器目录高速缓存。

**释放时间**

- 当应用程序进程终止时，释放应用程序目录高速缓存
- 当停止（db2stop）数据库管理器实例时，释放服务器目录高速缓存。

使用目录高速缓存可减小连接成本，因为这样消除了目录文件 I/O 并将检索目录信息所需的目录搜索减至最少。有两种类型的目录高速缓存:

- 分配并用于应用程序在其上运行的机器上的每个应用程序进程的应用程序目录高速缓存。
- 分配并用于一些内部数据库管理器进程的服务器目录高速缓存。

对于应用程序目录高速缓存，当应用程序发出它的第一个连接时，会读取每个目录文件且信息高速缓存在此应用程序的专用内存中。此高速缓存由应用程序进程用在后续连接请求上，并在应用程序进程的生命期内保持。如果未在应用程序目录高速缓存中发现数据库，那么将搜索目录文件以查找该信息，但是不更新高速缓存。如果应用程序修改目录条目，那么该应用程序中的下一个连接将导致刷新此应用程序的高速缓存。不会刷新其他应用程序的应用程序目录高速缓存。当应用程序进程终止时，释放高速缓存。（要刷新命令行处理器会话所使用的目录高速缓存，发出 db2 terminate 命令。）

对于服务器目录高速缓存，当启动（db2start）数据库管理器实例时，会读取每个目录文件并且将信息高速缓存在服务器内存中。将保留此高速缓存，直到该实例停止（db2stop）。如果此高速缓存中找不到某个目录条目，那么搜索目录文件以获取信息。在实例运行期间，从不刷新此服务器目录高速缓存。

**建议**: 如果目录文件不经常更改且性能很重要，那么使用目录高速缓存。

另外在远程客户机上，如果应用程序发出几个不同的连接请求，那么目录高速缓存会很有益。在这种情况下，高速缓存减少单个应用程序必须读取目录文件的次数。

目录高速缓存还可以提高获取数据库系统监视器快照的性能。另外，应显式引用快照调用中的数据库名称，而不使用数据库别名。

**注**：如果目录高速缓存打开并且在启动数据库管理器之后编目、取消编目、创建或删除数据库，那么在执行快照调用时可能会发生错误。

## discover - 发现方式

此参数确定客户机可以发出的发现请求的类型（如果发出请求）。

**配置类型**
数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**
可配置

**缺省值 [范围]**
SEARCH [DISABLE, KNOWN, SEARCH]

从客户机角度来看，将发生下列其中一种情况:

- 如果 *discover* = SEARCH，那么客户机可以发出搜索发现请求来查找网络上的 DB2 服务器系统。搜索发现提供了 KNOWN 发现所提供的功能的超集。如果 *discover* = SEARCH，那么客户机既可发出搜索发现请求也可发出已知发现请求。
- 如果 *discover* = KNOWN，那么客户机只能发出已知发现请求。通过对特定系统上的管理服务器指定一些连接信息，可以将 DB2 系统上的所有实例和数据库信息返回给客户机。
- 如果 *discover* = DISABLE，那么在客户机上禁用发现。

缺省发现方式为 SEARCH。

## discover_inst - 发现服务器实例

此参数指定 DB2 发现是否可以检测到此实例。

**配置类型**
数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**

ENABLE [ENABLE, DISABLE]

参数的缺省值 enable 指定可检测该实例，而 disable 则防止该实例被发现。

## fcm_num_buffers - FCM 缓冲区数

此参数指定数据库服务器之间及内部用于内部通信（消息）的 4 KB 缓冲区数。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

**32 位平台**

Automatic [128 - 65 300]

**64 位平台**

Automatic [128 - 524 288]

- 带有本地和远程客户机的数据库服务器: 缺省值为 1024
- 带有本地客户机的数据库服务器: 缺省值为 512
- 带有本地和远程客户机的分区数据库服务器: 缺省值为 4096

在单一分区数据库系统上，仅当通过将 *intra_parallel* 参数的值从缺省值 NO 更改为 YES 时才会启用分区内并行性。

可以同时设置初始值和 AUTOMATIC 属性。

当此参数设置为 AUTOMATIC 时，FCM 将监视资源使用情况，如果在 30 分钟内未使用资源，那么可以增加或减少资源。可以增加或减少的资源量取决于平台，特别是在 Linux 平台上，只能将资源在初始值基础上增加 25%。如果数据库管理器在实例启动时无法分配所指定的资源数，它就会逐渐减小配置值，直到可以启动实例为止。

如果同一机器上有多逻辑节点，那么您可能会发现需要增大此参数的值。如果由于系统上的用户数、系统上的数据库分区服务器数或这些应用程序的复杂程度而使消息缓冲区用尽，您可能也会发现需要增大此参数的值。

如果您正在使用多逻辑节点，那么同一机器上所有多逻辑节点将共享一个缓冲区数为 **fcm_num_buffers** 的池。通过 **fcm_num_buffers** 值的 N 次方（其中 N 为该物理机器上逻辑节点的数目）可以确定池大小。复查正在使用的值。考虑在多逻辑节点所在的机器上总共将分配多少 FCM 缓冲区。

如果您希望将 **fcm_num_buffers** 参数设置为特定值并带有 AUTOMATIC 属性，但不希望系统控制器线程将资源调整为低于所指定的值，那么您需要配置

**DB2_FCM_SETTINGS** 注册表变量。要启用此行为，请将 **DB2_FCM_SETTINGS** 注册表变量的 FCM_CFG_BASE_AS_FLOOR 选项设置为 YES（或者 TRUE）。将动态调整 **DB2_FCM_SETTINGS** 注册表变量值。

## fcm_num_channels - FCM 通道数

此参数指定用于每个数据库分区的 FCM 通道的数目。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器
- 带有本地客户机的卫星数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

**UNIX 32 位平台**

Automatic，起始值为 256，512，2 048 [128 - 120 000 ]

**UNIX 64 位平台**

Automatic，起始值为 256，512，2 048 [128 - 524 288 ]

**Windows 32 位**

Automatic，起始值为 10 000 [128 - 120 000 ]

**Windows 64 位**

Automatic，起始值为 256，512，2 048 [128 - 524 288 ]

- 对于带有本地客户机和远程客户机的数据库服务器，起始值为 512。
- 对于带有本地客户机的数据库服务器，起始值为 256。
- 对于带有本地和远程客户机的分区数据库环境服务器，起始值为 2 048。

在非分区数据库环境上，*intra_parallel* 参数必须是活动的，然后才能使用 *fcm_num_channels*。

FCM 通道表示在 DB2 引擎中运行的 EDU 之间的逻辑通信端点。控制流（请求和应答）和数据流（表队列数据）都依靠通道在各个分区之间传送数据。

当设置为 AUTOMATIC 时，FCM 将监视通道使用情况，并且会随要求的更改而逐渐分配和释放资源。

如果您希望将 **fcm_num_channels** 参数设置为特定值并带有 AUTOMATIC 属性，但不希望系统控制器线程将资源调整为低于所指定的值，那么您需要配置 **DB2_FCM_SETTINGS** 注册表变量。要启用此行为，请将 **DB2_FCM_SETTINGS** 注册表变量的 FCM_CFG_BASE_AS_FLOOR 选项设置为 YES（或者 TRUE）。将动态调整 **DB2_FCM_SETTINGS** 注册表变量值。

## fed_noauth - 绕过联合认证

此参数确定是否在实例中不会进行联合认证。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

No [ Yes; No ]

当 *fed_noauth* 设置为 *yes*、*authentication* 设置为 *server* 或 *server_encrypt*，并且 *federated* 设置为 *yes* 时，将绕过实例中的认证。假定认证将在数据源中进行。当 *fed_noauth* 设置为 *yes* 时，务必请小心谨慎地操作。既不在客户机上也不在 DB2 上执行认证。所有知道 SYSADM 认证名的用户都可对联合服务器使用 SYSADM 权限。

## federated - 联合数据库系统支持

此参数启用或禁用对落实分布式请求的应用程序的支持，那些请求是针对数据源（如 DB2 系列和 Oracle）管理的数据。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

No [ Yes; No ]

## federated_async - 每个查询的最大异步 TQ 数配置参数

此参数确定联合服务器支持的访问方案中的最大异步表队列（ATQ）数。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 启用联合时带有本地和远程客户机的分区数据库服务器。

**参数类型**

可联机配置

**缺省值 [范围]**

0 [0 到 32767（包括 0 和 32767 在内）、-1 和 ANY]

指定 ANY 或 -1 时，优化器将确定用于访问方案的 ATQ 数目。优化器将一个 ATQ 指定给方案中的所有合格 SHIP 或远程下推运算符。对 DB2_MAX_ASYNC_REQUESTS_PER_QUERY 服务器选项指定的值限制异步请求数。

**建议** *federated_async* 配置参数为专用寄存器和绑定选项提供缺省值或起始值。通过将 CURRENT FEDERATED ASYNCHRONY 专用寄存器、FEDERATED_ASYNCHRONY 绑定选项或 FEDERATED_ASYNCHRONY 预编译选项的值设置为一个较高或较低的数字，可以覆盖此参数的值。

如果专用寄存器或绑定选项不覆盖 *federated_async* 配置参数，那么该参数的值将确定联合服务器允许的访问方案中的最大 ATQ 数。如果专用寄存器或绑定选项覆盖此参数，那么专用寄存器或绑定选项的值确定方案中的最大 ATQ 数。

一旦当前工作单元落实后，对 *federated_async* 配置参数所作的任何更改就会影响动态语句。后续动态语句将自动识别新值。不需要重新启动联合数据库。当 *federated_async* 配置参数的值更改时，不会使嵌入式 SQL 程序包无效，也不会对它们进行隐式重新绑定。

如果您希望 *federated_async* 配置参数的新值影响静态 SQL 语句，那么需要重新绑定程序包。

## fenced_pool - 最大受防护进程数

对于线程化 db2fmp 进程（为线程安全存储过程和 UDF 提供服务的进程），此参数表示每个 db2fmp 进程中高速缓存的线程数目。对于非线程 db2fmp 进程，此参数表示高速缓存的线程数目。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

-1（*max_coordagents*），Automatic [-1; 0–64 000]

**计量单位**

计数器

**限制:**

- 如果此参数自动更新并且其值减小，那么数据库管理器不会主动终止 db2fmp 线程或进程，相反，它会在使用这些线程或进程时停止高速缓存它们，以便将高速缓存的 db2fmp 数目减小到新值。

- 如果此参数自动更新并且其值增大，那么数据库管理器将在创建 db2fmp 线程和进程时高速缓存更多线程和进程。
- 当此参数设置为缺省值 -1 时，它将采用 *max_coordagents* 配置参数的值。请注意，仅采用 *max_coordagents* 的值，不采用 Automatic 设置或行为。
- 当此参数设置为 AUTOMATIC 时（同样是缺省值）：
  - 数据库管理器允许根据协调代理程序的高水位标记增大高速缓存的 db2fmp 线程数和进程数。具体地说，此参数的自动行为允许它根据数据库管理器自启动后曾经同时注册的协调代理程序的最大数目增长。
  - 指定给此参数的值表示要高速缓存的 db2fmp 线程数和进程数的下限。

**建议**：如果您的环境使用受防护的存储过程或用户定义的函数，那么此参数可用来确保提供有适当数目的 db2fmp 进程，以处理在此实例上运行的最大数目的并发存储过程和 UDF，这将确保不需要创建新的设防方式进程作为存储过程和 UDF 执行的一部分。

如果由于提供给 db2fmp 进程的系统资源量不合适且影响数据库管理器的性能而导致缺省值对您的环境不适用，那么以下内容在作为提供调整该参数的起点方面可能有用:

```
fenced_pool = 允许其中 # 个应用程序同时进行存储过程和 UDF 调用
```

如果 *keepfenced* 设置为 YES，那么在高速缓存池中创建的每个 db2fmp 进程将继续存在并使用系统资源，即使在受防护的例程调用已处理并且返回至代理程序之后。

如果 *keepfenced* 设置为 NO，那么非线程 db2fmp 进程将在完成执行后终止，且没有任何高速缓存池。多线程 db2fmp 进程将继续存在，但是不会在这些进程中合用线程。这意味着即使将 *keepfenced* 设置为 NO，您也可以在系统上有一个线程化 C db2fmp 进程和一个线程化 Java db2fmp 进程。

在先前版本中，此参数称为 *maxdari*。

## group_plugin - 组插件

此参数指定组插件库的名称。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

Null [任何有效字符串]

缺省情况下，此值为 NULL，DB2 使用操作系统组查找。该插件将用于所有组查找。对于非 root 用户安装，如果使用了 DB2 用户标识和密码插件库，那么 db2rfe 命令必须在运行才能使用 DB2 产品。

## health_mon - 运行状况监视器

此参数允许您指定是否想要根据各种运行状况指示器来监视实例、它的相关数据库和数据库对象。

**配置类型**
数据库管理器

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
On [ On; Off ]

**相关参数**

如果打开了 *health_mon*（这是缺省情况），那么代理程序将收集关于所选对象的运行状况的信息。如果认为对象的运行状况不正常，那么根据您设置的阈值，可以发送通知，并且可以自动执行操作。如果关闭了 *health_mon*，那么不会监视对象的运行状况。

可以使用运行状况中心或 CLP 来选择想要监视的实例和数据库对象。还可以根据运行状况监视器收集的数据来指定应将通知发送至何处，以及执行哪些操作。

## indexrec - 索引重新创建时间

此参数指示数据库管理器将尝试重建无效索引的时间，以及在 DB2 前滚期间或在备用数据库上重放 HADR 日志期间是否重做任何索引构建。

**配置类型**
数据库和数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**

**UNIX 数据库管理器**
restart [ restart; restart_no_redo; access; access_no_redo ]

**Windows 数据库管理器**
restart [ restart; restart_no_redo; access; access_no_redo ]

**数据库** 使用系统设置 [system; restart; restart_no_redo; access; access_no_redo ]

此参数有五个可能的设置:

**SYSTEM**
在数据库管理器配置文件中指定的 *use system setting* 决定何时重建无效的索引，

以及在 DB2 前滚或 HADR 日志重放期间是否将重做任何索引构建日志记录。（注意: 此设置仅对数据库配置有效。）

**ACCESS**

第一次访问索引时将重建无效的索引。在 DB2 前滚或 HADR 日志重放期间将重做任何完全记录的索引构建。当启动了 HADR 且发生 HADR 接管时，将在接管之后第一次访问基础表时重建任何无效的索引。

**ACCESS_NO_REDO**

第一次访问基础表时将重建无效的索引。在 DB2 前滚或 HADR 日志重放期间将不重做任何完全记录的索引构建，那些索引将保留为无效状态。当启动了 HADR 且发生 HADR 接管时，将在接管之后第一次访问基础表时重建任何无效的索引。

**RESTART**

*indexrec* 的缺省值。将在显式或隐式发出 RESTART DATABASE 命令时重建无效的索引。在 DB2 前滚或 HADR 日志重放期间将重做任何完全记录的索引构建。当启动了 HADR 且发生 HADR 接管时，将在接管结束时重建任何无效的索引。

注意，如果启用 *autorestart* 参数，那么隐式发出 RESTART DATABASE 命令。

**RESTART_NO_REDO**

将在显式或隐式发出 RESTART DATABASE 命令时重建无效的索引。（如果启用了 *autorestart* 参数，将隐式地发出 RESTART DATABASE 命令。）在 DB2 前滚或 HADR 日志重放期间将不重做任何完全记录的索引构建，而是在前滚完成时或在 HDAR 接管发生时将重建那些索引。

当出现严重的磁盘问题时，索引可能会变为无效。如果数据本身出现这个问题，那么数据可能丢失。但是，如果索引出现这个问题，那么可通过重新创建该索引来恢复索引。如果在用户连接至数据库时重建索引，那么可能出现两个问题:

- 重新创建索引文件时可能会发生响应时间意外降低现象。访问表和使用此特定索引的用户将在重建索引时等待。
- 在重新创建索引之后可能挂起意外的锁定，尤其是导致索引重新创建的用户事务从未执行过 COMMIT 或 ROLLBACK 的情况下更是如此。

**建议**: 在高端用户服务器上且如果重新启动时间不重要，那么此选项的最佳选择将是在 DATABASE RESTART 时重建该索引，以作为在崩溃后重新将该数据库联机的过程的一部分。

将此参数设置为“ACCESS”或“ACCESS_NO_REDO”将导致重新创建索引时数据库管理器的性能降级。任何访问该特定索引或表的用户将必须等待，直到重新创建索引为止。

如果将此参数设置为“RESTART”，那么重新启动数据库所花的时间将因重新创建索引而较长，但是一旦数据库恢复联机，正常处理将不受影响。

**注**: 在进行数据库恢复时，将除去可在属于正在恢复的数据库的文件系统上执行的所有 SQL 过程。如果 *indexrec* 设置为 RESTART，那么会从数据库目录中抽取所有 SQL 过程可执行文件，并在下一次连接至数据库时放回到该文件系统上。如果未将 *indexrec* 设置为 RESTART，那么仅在第一次执行该 SQL 过程时才会将 SQL 可执行文件抽取到该文件系统上。

仅当对索引构建操作（如 CREATE INDEX 和 REORG INDEX 操作）或对索引重建操作激活了完全日志记录时，RESTART 与 RESTART_NO_REDO 值或 ACCESS 与 ACCESS_NO_REDO 值之间的差异才会很明显。可以通过启用 *logindexbuild* 数据库配置参数或者通过在改变表时启用 LOG INDEX BUILD 来激活日志记录。通过将 *indexrec* 设置为 RESTART 或 ACCESS，就可以前滚涉及记录的索引构建的操作而不会使索引对象处于无效状态，如果索引对象处于无效状态，那么需要以后重建该索引。

## instance_memory - 实例内存

如果您正在使用的 DB2 数据库产品具有内存使用量限制或者您将内存使用量设置为特定值，那么此参数指定可以为数据库分区分配的最大内存量。否则，AUTOMATIC 设置将允许实例内存根据需要而增加。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

**32 位平台**

Automatic [0 - 1 000 000]

**64 位平台**

Automatic [0 - 68 719 476 736]

**DB2 Express® Edition和 DB2 Express-C**

Automatic [0 - 1048576]

**DB2 Workgroup Server Edition**

Automatic [0 - 4194304]

**计量单位**

页 (4 KB)

**分配时间**

不适用

**释放时间**

不适用

**instance_memory** 的缺省值为 AUTOMATIC。AUTOMATIC 设置将导致在激活数据库分区时计算值。计算值介于系统上的物理 RAM 的 75% 到 95% 之间 - 系统越大，此百分比越高。对于具有内存使用量限制的 DB2 数据库产品，计算值也会受到产品许可证允许的最大值的限制。对于具有多个逻辑数据库分区的数据库分区服务器，此计算值是除以逻辑数据库分区数而获得的值。

从版本 9.5 修订包 5 开始，对于没有内存使用量限制的 DB2 数据库产品，AUTOMATIC 设置的计算值不会对实例中分配的内存强制施加限制。对于版本 9.5 修订包 4 或更低版本，AUTOMATIC 设置的计算值表示对于所有 DB2 数据库产品的限制。

### 动态更新 instance_memory

- 要动态更新 **instance_memory**，需要连接实例。有关详细信息，请参阅 ATTACH 命令。
- 对于具有内存使用量限制的 DB2 数据库产品，要动态更新 **instance_memory**，必须指定一个小于任何许可证限制的值，或者指定 AUTOMATIC。否则，更新将失败并且返回 SQL5130N 错误消息。
- 要动态更新 **instance_memory**，必须指定一个小于物理 RAM 量的值，或者指定 AUTOMATIC。否则，将延迟到发出下一个 db2start 命令时才会更新，并且将返回 SQL1362W 警告消息。
- 要动态更新 **instance_memory**，必须指定一个大于当前正在使用的实例内存量的值。否则，将延迟到重新启动该实例时才会更新，并且将返回 SQL1362W 警告消息。通过将 db2pd -dbptnmem 命令输出中的*当前使用量*值减去*已高速缓存的内存*值，就可以确定正在使用的实例内存量。最小值将为所有数据库分区中正在使用的最大实例内存量。
- 如果 **instance_memory** 设置为一个大于物理 RAM 量的值，那么您发出的下一个 db2start 命令将失败，并且将返回 SQL1220N 错误消息。
- 如果将 **instance_memory** 动态更新为 AUTOMATIC，那么将立即重新计算值。

### 对于 DPF 实例的限制

在 DPF 实例中，不应对 **instance_memory** 使用特定值。在 DPF 实例中，建议不要对 **instance_memory** 使用特定值，这是因为 **instance_memory** 是一个数据库管理器配置参数，不能为不同的数据库分区指定不同的值。因此，很难建立一个适合于所有数据库分区的设置，因为这些数据库分区可能具有不同的内存需求。

### 控制 DB2 内存消耗

根据工作负载和配置不同，DB2 内存消耗也将不同。除此之外，如果启用了自调整 **database_memory**，它也会影响 DB2 内存消耗。当 **database_memory** 设置为 AUTOMATIC，并且自调整内存管理器 (STMM) 处于活动状态时，就启用了自调整 **database_memory**。

如果实例正在一个没有内存使用量限制的 DB2 数据库产品上运行，并且 **instance_memory** 设置为 AUTOMATIC，那么不会强制施加 **instance_memory** 限制。数据库管理器将根据需要来分配系统内存。如果启用了自调整 **database_memory**，那么 STMM 将更新配置以获得最佳性能，同时还会监视可用系统内存。通过监视可用内存，可确保不会过量使用系统内存。

如果实例正在一个具有内存使用量限制的 DB2 数据库产品上运行，或者 **instance_memory** 设置为特定值，那么将强制施加 **instance_memory** 限制。数据库管理器分配的系统内存最多只能为此限制；当达到此限制时，应用程序会接收到内存分配错误。另外还有一些注意事项，如下所示:

- 如果启用了自调整 **database_memory**，并且 **instance_memory** 设置为特定值，那么 STMM 将更新配置以获得最佳性能，同时还会保持足够的可用实例内存。这可确保有足够的实例内存可用，以满足易失内存需求。不会监视系统内存。
- 如果启用了自调整 **database_memory**，并且 **instance_memory** 设置为 AUTOMATIC，那么在这种情况下将对具有内存使用量限制的 DB2 数据库产

品强制施加 **instance_memory** 限制，并且 STMM 将更新配置以获得最佳性能，同时还会监视可用系统内存和保持足够的可用实例内存。

**监视实例内存使用量**

使用 db2pd -dbptnmem 命令来显示有关实例内存使用量的详细信息。

使用新的 ADMIN_GET_DBP_MEM_USAGE 表函数来获取特定数据库分区或所有数据库分区的 DB2 实例耗用的内存总量。此表函数还会返回当前上限值。

分配快速通信管理器（FCM）共享内存时，将在每个本地数据库分区的 **instance_memory**使用中说明该数据库分区在系统的总 FCM 共享内存大小中所占的份额。

## intra_parallel - 启用分区内并行性

此参数指定数据库管理器是否可以使用分区内并行性。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

NO (0) [SYSTEM (-1), NO (0), YES (1)]

值 -1 导致该参数值设置为“YES”或“NO”，这取决于正在运行数据库管理器的硬件。

当此参数设置为“YES”时，一些参数可利用并行性能改进，这些操作包括数据库查询和索引创建。

**注:**

- 并行索引创建不使用此配置参数。
- 如果更改此参数值，那么可能将程序包重新绑定至数据库，并且可能会使性能有一定下降。

## java_heap_sz - 最大 Java 解释器堆大小

此参数确定由已启动以便为 Java DB2 存储过程和 UDF 提供服务的 Java 解释器使用的堆的最大大小。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器

- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

**HP-UX**

4096 [0 - 524 288]

**所有其他操作系统**

2048 [0 - 524 288]

**计量单位**

页 (4 KB)

**分配时间**

当 Java 存储过程或 UDF 启动时

**释放时间**

当受防护的 db2fmp 进程或可信的 db2agent 进程终止时。

每个 DB2 进程都有一个堆（一个用于 Linux 和 UNIX 平台上的代理程序或子代理程序，一个用于其他平台上的每个实例）。每个受防护的 UDF 和受防护的存储过程进程都有一个堆。可信的例程的每个代理程序（不包括子代理程序）都有一个堆。每个运行 Java 存储过程的 db2fmp 进程都有一个堆。对于多线程 db2fmp 进程，使用线程安全受防护的例程的多个应用程序由单个堆提供服务。在所有情况下，只有运行 Java UDF 或存储过程的代理程序或进程才分配此内存。在分区数据库系统上，在各数据库分区使用同一个值。

将 XML 数据作为 IN、OUT 或 INOUT 参数传递至存储过程时，将具体化该数据。如果使用的是 Java 存储过程，那么可能需要根据 XML 参数的数量和大小以及正在同时执行的外部存储过程的数目来增大堆大小。

## jdk_path - Java 软件开发者工具箱安装路径

此参数指定要用于运行 Java 存储过程和用户定义的函数的 Java 软件开发者工具箱 (SDK) 的安装目录。Java 解释器使用的 CLASSPATH 和其他环境变量是通过此参数的值计算的。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

Null [有效路径]

如果 Java SDK 是随 DB2 产品一起安装的，那么此参数设置正确。但是，如果复位数据库管理器 (dbm cfg) 参数，那么需要指定 Java SDK 的安装位置。

## keepfenced - 保留受防护进程

此参数指示在设防方式例程调用完成后，是否保留设防方式进程。将设防方式进程创建为独立的系统实体以便将用户编写的设防方式代码与数据库管理器代理程序分开。此参数仅可应用于数据库服务器。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

Yes [ Yes; No ]

如果将 *keepfenced* 设置为 *no*，且正执行的例程不是线程安全的，那么对于每个设防方式调用，新的设防方式都会被创建并被破坏。如果将 *keepfenced* 设置为 *no*，且正在执行的例程是线程安全的，那么设防方式进程仍然存在，但是为调用创建的线程将会终止。如果将 *keepfenced* 设置为 *yes*，那么对于后续设防方式调用，将重复使用设防方式进程或线程。当数据库管理器停止时，将终止所有未完成的设防方式进程和线程。

将此参数设置为 *yes* 将导致每个激活的设防方式进程的数据库管理器消耗更多系统资源，最多为在 *fenced_pool* 参数中包含的值。仅当没有提供现有设防方式进程来处理后续受防护的例程调用时，才会创建新的进程。如果将 *fenced_pool* 设置为 0，那么将忽略此参数。

**建议**：如果环境中的设防方式请求数目相对于不设防方式请求数目来说很大，并且系统资源不受约束，那么可将此参数设置为 *yes*。这将通过避免初始设防方式进程创建开销来提高设防方式进程性能，因为将使用现有设防方式进程来处理调用。尤其是对于 Java 例程，这将节省启动“Java 虚拟机”（JVM）的开销，从而使性能有很显著的提高。

例如，在 OLTP 信贷银行事务应用程序中，用来执行每个事务的代码可以在设防方式进程中执行的存储过程中执行。在此应用程序中，主要的工作负载在设防方式进程之外执行。如果将此参数设置为 *no*，每个事务都会产生创建新的设防方式进程的开销，这将使性能显著降低。但是，如果此参数设置为 *yes*，那么每个事务都将尝试使用现有设防方式进程，这将避免此开销。

在先前版本的 DB2 中，此参数称为 *keepdari*。

## local_gssplugin - 用于实例级本地授权的 GSS API 插件

此参数指定当 *authentication* 数据库管理器配置参数的值设置为 GSSPLUGIN 或 GSS_SERVER_ENCRYPT 时要用于实例级本地授权的缺省 GSS API 插件库的名称。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器

- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

Null [任何有效字符串]

## max_connections - 最大客户机连接数

此参数指示每个数据库分区允许的最大客户机连接数。

**配置类型**

数据库管理器

**参数类型**

可联机配置

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的数据库服务器或连接服务器（适用于 *max_connections*、*max_coordagents*、*num_initagents* 和 *num_poolagents*，如果使用的是联合环境，那么还适用于 *federated_async*）

**缺省值 [范围]**

-1 和 AUTOMATIC（*max_coordagents*）[-1 和 AUTOMATIC; 1–64000 ]

设置 -1 表示将使用与 *max_coordagents* 关联的值，而不使用自动设置或行为。AUTOMATIC 意味着数据库管理器将使用最佳技术来选取此参数的值。AUTOMATIC 是配置文件中的一个 ON/OFF 开关，它与值无关，因此 -1 和 AUTOMATIC 都可以是缺省设置。

有关详细信息，请参阅：第 425 页的『配置 max_coordagents 和 max_connections 时的限制和行为』。

### 集中器

当 *max_connections* 等于或小于 *max_coordagents* 时，集中器关闭。当 *max_connections* 大于 *max_coordagents* 时，集中器将打开。

此参数控制可以与实例中的数据库分区连接的最大客户机应用程序数。通常，每个应用程序都被指定了一个协调代理程序。代理程序简化了应用程序与数据库之间的操作。当使用此参数的缺省值时，将不激活集中器功能部件。因此，每个代理程序都在它自己的专用内存中运行，并与其他代理程序共享数据库管理器和数据库全局资源，如缓冲池。将此参数设置为大于缺省值的值时，会激活集中器功能部件。

## max_connretries - 节点连接重试次数

此参数指定尝试在两个数据库分区服务器之间建立 TCP/IP 连接的最大次数。

**配置类型**

数据库管理器

**适用于** 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

5 [0–100]

如果试图在两个数据库分区服务器之间建立通信失败（例如，达到 *conn_elapse* 参数指定的值），那么 *max_connretries* 指定可对数据库分区服务器进行连接重试的次数。如果超过为此参数指定的值，将返回一个错误。

## max_coordagents - 最大协调代理程序数

此参数用来限制协调代理程序数。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

200, Automatic [-1; 0–64 000]

设置 -1 将转换为值 200。

有关详细信息，请参阅：第 425 页的『配置 max_coordagents 和 max_connections 时的限制和行为』。

### 集中器

当集中器关闭时（即，当 *max_connections* 等于或小于 *max_coordagents* 时），此参数确定服务器节点上可同时存在的协调代理程序的最大数目。

连接至数据库或连接至实例的每个本地或远程应用程序各获得一个协调代理程序。需要实例连接的请求包括 CREATE DATABASE、DROP DATABASE 和“数据库系统监视器”命令。

当集中器打开时（即，*max_connections* 大于 *max_coordagents* 时），可能有比协调代理程序多的连接来为它们提供服务。仅当有协调代理程序在为应用程序提供服务时，应用程序才处于活动状态。否则，应用程序处于不活动状态。来自活动应用程序的请求将由数据库协调代理程序（以及 SMP 或 MPP 配置中的子代理程序）来提供服务。来自不活动的应用程序的请求将会进行排队，直到指定数据库协调代理程序来为该应用程序提供服务，此时，应用程序变成活动的。因此，此参数可用来控制系统上的负载。

## max_querydegree - 最大查询并行度

此参数指定用于在数据库管理器的此实例上执行的任何 SQL 语句的最大分区内并行度。当执行某条 SQL 语句时，该语句在一个数据库分区内使用的并行操作的数目将不大于此数目。

**配置类型**
数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**
可联机配置

**传播类** 语句边界

**缺省值 [范围]**
-1 (ANY) [ANY, 1 - 32767] （ANY 表示由系统确定）

必须将 **intra_parallel** 配置参数设置为 YES，以允许数据库分区将分区内并行性用于 SQL 语句。创建并行索引不再需要 **intra_parallel** 参数。

此配置参数的缺省值为 -1。此值表示系统使用优化器确定的并行度；否则，使用用户指定的值。

**注:** 在编译语句时，可使用 CURRENT DEGREE 专用寄存器或 **DEGREE** 绑定选项指定 SQL 语句的并行度。

可以使用 SET RUNTIME DEGREE 命令来修改活动应用程序的最大查询并行度。实际使用的运行时并行度是下列值中较小的那一个:

- **max_querydegree** 配置参数
- 应用程序运行时并行度
- SQL 语句编译并行度

此配置参数仅适用于查询。

## max_time_diff - 节点间的最大时差

此参数以分钟为单位指定列示在节点配置文件文件中的数据库分区服务器所允许的最大时间差。

**配置类型**
数据库管理器

**适用于** 带有本地和远程客户机的分区数据库服务器

**参数类型**
可配置

**缺省值 [范围]**
60 [1 - 1 440]

**计量单位**

分钟

每个数据库分区服务器都有自己的系统时钟。如果两个或多个数据库分区服务器与某一事务相关联，并且它们的时钟之间的时间差超过了由 MAX_TIME_DIFF 参数所指定的时间量，那么该事务将被拒绝，并且返回 SQLCODE。（仅当数据修改与事务相关时，才拒绝事务。）

还可以在数据库分区环境中返回 SQLCODE；在这些环境中，DB2 会将系统时钟与保存到 SQLOGCTL.LFH 日志控制文件中的虚拟时间戳记（VTS）进行比较。如果 .LFH 文件中的时间戳记小于系统时间，那么会将数据库日志中的时间设置为 VTS，直到系统时钟与此时间相匹配为止。尽管多个节点之间的时间差小于 MAX_TIME_DIFF 参数值，也将返回 SQL1473N 错误消息。

DB2 使用*全球标准时间*（UTC），因此在设置此参数时不考虑不同的时区。“全球标准时间”与“格林威治标准时间”相同。

## maxagents - 最大代理程序数

在版本 9.5 中不推荐使用此参数，但是版本 9.5 之前的数据服务器和客户机仍然使用此参数。DB2 V9.5 数据库管理器将忽略对此配置参数指定的任何值。

**注:** 下列信息仅适用于版本 9.5 之前的数据服务器和客户机。

此参数指示可在任何给定时间接受应用程序请求的数据库管理器代理程序（无论是协调代理程序还是子代理程序）的最大数目。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

200 [1 - 64 000]

在带有本地和远程客户机的分区数据库服务器上为 400® [1 - 64 000]

**计量单位**

计数器

如果您想限制协调代理程序数，请使用 *max_coordagents* 参数。

此参数可在内存受约束的环境中来限制数据库管理器使用的内存总量，因为每个附加代理程序都需要附加内存。

**建议:** *maxagents* 的值至少应为每个数据库中允许同时访问的 *maxappls* 的值之和。如果数据库数大于 *numdb* 参数，那么最安全的过程是使用具有 *maxappls* 的最大值的 *numdb* 产品。

每个附加代理程序都需要一些在数据库管理器启动时分配的资源开销。

如果在尝试连接至数据库时遇到内存错误，请尝试进行下列配置调整:

- 在未启用查询内并行性的非分区数据库环境中，增大 *maxagents* 数据库配置参数的值。
- 在分区数据库环境或启用了查询内并行性的环境中，增大 *maxagents* 或 *max_coordagents* 中较大者的值。

## maxcagents - 最大并发代理程序数

在版本 9.5 中不推荐使用此参数，但是版本 9.5 之前的数据服务器和客户机仍然使用此参数。DB2 V9.5 数据库管理器将忽略对此配置参数指定的任何值。

**注:** 下列信息仅适用于版本 9.5 之前的数据服务器和客户机。

通过限制可同时执行一个数据库管理器事务的数据库管理器代理程序的最大数目，此参数可用来控制并行应用程序活动高峰期内系统上的负载。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

-1 (*max_coordagents*) [-1; 1 – *max_coordagents* ]

**计量单位**

计数器

此参数不限制可与数据库连接的应用程序的数目。此参数只限制在任何时间数据库管理器可同时处理的数据库管理器代理程序的数目，从而限制在处理高峰期内系统资源的使用。例如，可以让一个系统需要大量连接但是只将有限内存量用于这些连接。在并行活动高峰期可能会导致过度的操作系统页面调度的环境中调整此参数可能会很有用。

值 -1 表示限制为 *max_coordagents*。

**建议:** 在大多数情况下，此参数的缺省值将是可接受的。当应用程序的高并行性导致问题时，可以使用基准程序测试来调整此参数以优化数据库的性能。

## mon_heap_sz -“数据库系统监视器堆大小”

此参数确定分配给数据库系统监视器数据的内存量（以页计）。当进行数据库监视活动（例如，生成快照、调整监视器开关、重新设置监视器或激活事件监视器）时，从监视器堆分配内存。

对于版本 9.5，此数据库配置参数的缺省值为 AUTOMATIC，这表示监视器堆可以根据需要增大，直到达到 *instance_memory* 限制。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

Automatic [0 - 60 000]

**计量单位**

页 (4 KB)

**分配时间**

当使用 db2start 命令启动数据库管理器时

**释放时间**

当使用 db2stop 命令停止数据库管理器时

如果值为零，那么将阻止数据库管理器收集数据库系统监视器数据。

**建议**：监视活动所需的内存量取决于监视应用程序（捕获快照的应用程序或事件监视器）的数目、设置了哪些开关以及数据库活动的级别。

如果设置为 AUTOMATIC，那么在此堆中配置的内存都用尽，且在实例共享内存区域中没有更多的非保留内存的情况下，将会发生以下情况:

- 当第一个应用程序连接至定义了此事件监视器的数据库时，会将一条错误消息写入到管理通知日志。
- 如果使用 SET EVENT MONITOR 语句动态启动的事件监视器失败，那么向您的应用程序返回错误代码。
- 如果一个监视器命令或 API 子例程失败，那么向您的应用程序返回错误代码。

如果设置为数字值（非 AUTOMATIC），那么不超过配置的内存量可供监视（与可用实例内存量无关）。

## nodetype - 机器节点类型

此参数提供关于已安装在机器上的 DB2 产品的信息，从而提供关于数据库管理器配置类型的信息。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器

- 带有本地和远程客户机的分区数据库服务器

**参数类型**

参考

以下是此参数返回的可能值以及与该节点类型相关的产品:

- **带有本地和远程客户机的数据库服务器** - 一个 DB2 服务器产品，它支持本地和远程数据服务器运行时客户机，并能访问其他远程数据库服务器。
- **客户机** - 能够访问远程数据库服务器的数据服务器运行时客户机。
- **带有本地客户机的数据库服务器** - 一个 DB2 关系数据库管理系统，它支持本地数据服务器运行时客户机，并且能够访问其他远程数据库服务器。
- **带有本地和远程客户机的分区数据库服务器** - 一个 DB2 服务器产品，它支持本地和远程数据服务器运行时客户机，能够访问其他远程数据库服务器，并且能够实现并行性。

## notifylevel - 通知级别

此参数指定写入管理通知日志的管理通知消息的类型。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

3 [ 0 — 4 ]

在 Linux 和 UNIX 平台上，管理通知日志是称为 *instance*.nfy 的文本文件。在 Windows 上，所有管理通知消息都写到“事件日志”中。错误可由 DB2、“运行状况监视器”、Capture 和 Apply 程序以及用户应用程序写入。

此参数的有效值为:

- **0** - 未捕获任何管理通知消息。（不建议使用此设置。）
- **1** - 致命或不可恢复错误。仅记录致命和不可恢复错误。要从这些情况中的某些情况进行恢复，可能需要来自 DB2 服务机构的帮助。
- **2** - 需要立即操作。记录了需要来自系统管理员或数据库管理员立即注意的情况。如果未解决该情况，那么它可能会导致致命错误。非常重要并且没有错误的活动（例如，恢复）的通知可能也在此级别记录。此级别将捕获“运行状况监视器”警报。
- **3** - 重要信息，不需要立即操作。记录没有威胁也不需要立即操作但是可能表示并非最佳系统的情况。此级别将捕获“运行状况监视器”警报和“运行状况监视器”警告和“运行状况监视器”引起注意信息。
- **4** - 参考消息。

管理通知日志包括具有最多且包括 *notifylevel* 的值的消息。例如，将 *notifylevel* 设置为 3 将导致管理通知日志包括可应用于级别 1、2 和 3 的消息。

要使用户应用程序能够写入通知文件或“Windows 事件日志”，它必须调用 db2AdminMsgWrite API。

**建议**：您可能希望增大此参数的值以收集更多问题确定数据来帮助解决问题。注意，您必须将 *notifylevel* 的值设置为 2 或更高，以便“运行状况监视器”向在其配置中定义的联系人发送所有通知。

## num_initagents - 池中的初始代理程序数

此参数确定在 DB2START 时在代理程序池中创建的初始空闲代理程序数。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

0 [0–64 000]

数据库管理器始终在 db2start 命令期间启动 *num_initagents* 个空闲代理程序，但在启动期间此参数的值大于 *num_poolagents* 并且 *num_poolagents* 未设置为 AUTOMATIC 时除外。在这种情况下，数据库管理器仅启动 num_poolagents 个空闲代理程序，这是因为没有理由启动比可以合用的空闲代理程序数更多的空闲代理程序。

## num_initfenced - 受防护进程的初始数目

此参数指示在 DB2START 时间 db2fmp 池中创建的非线程的空闲 db2fmp 进程的初始数目。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

0 [0–64 000]

设置此参数将减少运行非线程安全 C 和 Cobol 例程的初始启动时间。如果未指定 *keepfenced*，那么将忽略此参数。

为您的系统将 *fenced_pool* 设置为适当的大小比在 DB2START 时间启动一些 db2fmp 进程重要得多。

在先前版本中，此参数称为 *num_initdaris*。

## num_poolagents - 代理程序池大小

此参数设置空闲代理程序池的最大大小。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值** 100，Automatic [-1，0–64 000]

此配置参数设置为 AUTOMATIC 并且缺省值为 100。设置 -1 仍受支持，并且它会转换为值 100。当此参数设置为 AUTOMATIC 时，数据库管理器将自动管理池中的空闲代理程序数。通常，这表示在代理程序完成其工作后，它不会终止，而是空闲一段时间。根据代理程序的工作负载和类型，它可以在某个时间段后终止。

使用 AUTOMATIC 时，仍可以指定 num_poolagents 配置参数的值。当前合用的空闲代理程序数小于或等于指定的值时，始终会合用其他空闲代理程序。

**示例:**

*num_poolagents* **设置为 100 和 AUTOMATIC**

在代理程序变得可用后，将它添加到空闲代理程序池中，数据库管理器会在某个时刻评估是否应将其终止。在数据库管理器考虑终止代理程序时，如果合用的空闲代理程序总数大于 100，那么将终止此代理程序。如果空闲代理程序数小于 100，那么空闲代理程序将保持等待工作。使用 AUTOMATIC 设置允许合用超过 100 的其他空闲代理程序，在具有大量系统活动期间，当工作频率在一个较大的范围波动时，这样做很有用。对于在任何给定时间空闲代理程序数可能会小于 100 的情况，保证合用代理程序。由于新工作产生较少的启动成本，所以在具有较少系统活动期间合用代理程序可以获得好处。

**动态配置** *num_poolagents*

如果该参数值增大到大于合用的代理程序数，那么立即就会产生效果。在新代理程序变得空闲时，将合用这些程序。如果该参数值减小，那么数据库管理器不会立即减少池中的代理程序数。更确切地说，池大小将保持不变，并且在使用代理程序时终止它们以使它们再次变得空闲 - 这样逐渐将池中的代理程序数减小到新限制。

**建议**：对于大多数环境来说，使用缺省值 0 和 AUTOMATIC 就可以了。如果您感觉正在创建和终止太多代理程序，那么在这种特定工作负载下，可以考虑增大 *num_poolagents* 的值，并同时使参数保持设置为 AUTOMATIC。

## numdb - 同时处于活动状态的数据库（包括主机和 System i 数据库）的最大数目

此参数指定可以同时处于活动状态（即，具有与它们相连接的应用程序）的本地数据库数或者是可以在 DB2 Connect 服务器上编目的不同数据库别名的最大数目。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

**UNIX** 8 [1 — 256 ]

**带有本地和远程客户机的 Windows 数据库服务器**

8 [1 — 256 ]

**带有本地客户机的 Windows 数据库服务器**

3 [1 — 256 ]

**计量单位**

计数器

每个数据库都占用存储器，并且活动数据库使用新的共享内存段。

**建议**：一般情况下，最好将此值设置为已经为数据库管理器定义的数据库的实际数目，并将此值增大大约 10% 以满足将来的增长需要。

更改 *numdb* 参数可能会影响已分配的总内存量。因此，建议不要频繁更新此参数。当更新此参数时，应考虑可为数据库或连接至该数据库的应用程序分配内存的其他配置参数。

## query_heap_sz - 查询堆大小

在版本 9.5 中不推荐使用此参数，但是版本 9.5 之前的数据服务器和客户机仍然使用此参数。DB2 V9.5 数据库管理器将忽略对此配置参数指定的任何值。

**注**：下列信息仅适用于版本 9.5 之前的数据服务器和客户机。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

1 000 [2 - 524 288 ]

**计量单位**

页 (4 KB)

**分配时间**

当应用程序（以本地方式或远程方式）与该数据库连接时

**释放时间**

当应用程序与数据库断开连接时，或与实例拆离时

此参数指定可为查询堆分配的**最大**内存量，以便确保应用程序不会不必要地消耗代理程序中的大量虚拟内存。

查询堆用来将每个查询存储在代理程序专用内存中。每个查询的信息由输入和输出 SQLDA、语句文本、SQLCA、程序包名、创建程序、节号以及一致性标记组成。

查询堆也用于分配给分块游标的内存。此内存由游标控制块和全分辨输出 SQLDA 组成。

所分配的初始查询堆将与该应用程序支持层堆的大小相同，后者由 *aslheapsz* 参数指定。该查询堆大小必须大于或等于二（2），且必须大于或等于 *aslheapsz* 参数。如果此查询堆不够大，无法处理给定的请求，将重新分配它，使其达到该请求所需的大小（不超过 *query_heap_sz*）。如果此新查询堆超过 *aslheapsz* 的 1.5 倍，那么在该查询结束时将重新分配该查询堆，使其大小为 *aslheapsz*。

**建议**：在大多数情况下，缺省值已足够使用。作为最小值，您应该将 *query_heap_sz* 设置为至少大于 *aslheapsz* 五倍的值。这将允许进行超过 *aslheapsz* 的查询，并为要在同一个给定时间打开的三个或四个分块游标提供附加内存。

如果您拥有很大的 LOB，那么可能需要增大此参数的值，以便查询堆的大小足够容纳那些 LOB。

## release - 配置文件发行版级别

此参数指定配置文件的发行版级别。

**配置类型**

数据库管理器，数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

参考

## resync_interval - 事务再同步时间间隔

此参数指定事务管理器（TM）、资源管理器（RM）或同步点管理器（SPM）重试恢复 TM、RM 或 SPM 中任何未完成的不确定事务的时间间隔（以秒计）。当有事务运行于分布式工作单元（DUOW）环境中时此参数适用。此参数也适用于联合数据库系统的恢复。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

180 [1 - 60 000 ]

**计量单位**

秒

**建议**: 如果在您的环境中不确定事务将不干扰应用于您的数据库的其他事务，那么您可能希望增大此参数的值。如果正在使用 DB2 Connect 网关访问 DRDA2 应用程序服务器，那么即使将不干扰本地数据访问，也应考虑不确定事务对该应用程序服务器可能产生的影响。如果没有不确定事务，那么性能影响将最小。

## rqrioblk - 客户机 I/O 块大小

此参数指定数据库服务器上远程应用程序及其数据库代理程序之间的通信缓冲区大小。它还用来确定在打开分块游标时数据服务器运行时客户机中的 I/O 块大小。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

32 767 [4 096 - 65 535 ]

**计量单位**

字节

**分配时间**

- 当远程客户机应用程序对服务器数据库发出连接请求时
- 打开分块游标时，在客户机中打开其他块期间

**释放时间**

- 当远程应用程序与服务器数据库断开连接时
- 当关闭分块游标时

当数据服务器运行时客户机请求与远程数据库连接时，将在客户机上分配此通信缓冲区。在数据库服务器上，最初分配一个大小为 32 767 字节的通信缓冲区，直到建立连接从而服务器可确定客户机中 *rqrioblk* 的值为止。服务器知道此值后，如果客户机缓冲区不为 32 767 字节，那么服务器将重新分配其通信缓冲区。

用于分块游标的内存是在应用程序专用地址空间之外分配的，所以应确定要分配给每个应用程序的最佳专用内存量。如果数据服务器运行时客户机不能为分块游标分配应用程序的专用内存之外的空间，那么将打开非分块游标。

**建议**：对于非分块游标，增大此参数的值的一个原因是：要由单个查询语句发送的数据（例如，大对象数据）很大，以致缺省值不够。

还应考虑此参数对分块游标的数目和潜在大小的影响。如果所传送的行的数目或大小较大（例如，如果数据量大于 4096 个字节），那么大行块可能获得更佳性能。但是，由于较大的记录块会增大每个连接的工作集内存大小，所以要加以折衷。

大记录块还可能导致比应用程序实际需要的更多的访存请求。可通过在应用程序中的 SELECT 语句上使用 OPTIMIZE FOR 子句来控制访存请求数。

## sheapthres -“排序堆阈值”

此参数是对专用排序在任何给定时间可以使用的总内存量的实例范围软限制。当某个实例使用的专用排序内存总量达到此限制时，为其他传入专用排序请求分配的内存将显著减少。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器
- OLAP 函数

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

**UNIX 32 位平台**

0 [0, 250 - 2097152]

**Windows 32 位平台**

0 [0, 250 - 2097152]

**64 位平台**

0 [0, 250 - 2147483647]

**计量单位**

页 (4 KB)

使用排序堆的操作示例包括: 排序、散列连接、动态位映射（用于索引 AND 运算和星型连接）以及表位于内存中的操作。

显式定义阈值可防止数据库管理器对大量排序使用过量内存。

当从非分区数据库移至分区数据库环境时，不应增大此参数的值。一旦在单个数据库分区环境中调整了数据库和数据库管理器配置参数，在大多数情况下，相同的值在分区数据库环境将同样合适。将此参数设置为在不同节点或数据库分区上具有不同值的唯一方法是创建多个 DB2 实例。这将需要通过不同数据库分区组管理不同的 DB2 数据库。这种调度将无法发挥分区数据库环境的许多优点。

当实例级 *sheapthres* 设置为 0 时，仅在数据库级进行排序内存使用量跟踪，排序内存分配受数据库级 *sheapthres_shr* 配置参数值约束。

仅当数据库管理器配置参数 *sheapthres* 设置为 0 时，才允许自动调整 *sheapthres_shr*。

如果下列任何条件成立，此参数就不能动态更新:

- *sheapthres* 的起始值是 0，目标值不是 0。
- *sheapthres* 的起始值不是 0，目标值是 0。

**建议**: 理想情况下，应将此参数设置为您在数据库管理器实例中拥有的最大 *sortheap* 参数的一个合理倍数。此参数**至少**应是该实例中为任何数据库定义的最大 *sortheap* 的两倍。

如果您正执行专用排序并且您的系统不受内存约束，那么可使用下列步骤来计算此参数的理想值:

1. 计算用于每个数据库的典型排序堆大小:

   ```
   （对该数据库运行的并发代理程序的典型数目）
   * （为该数据库定义的 sortheap）
   ```

2. 对以上结果求和，该值提供可在实例中所有数据库的典型环境中使用的总排序堆。

应使用基准程序技术来调整此参数以找到在排序性能和内存使用之间的相应平衡。

可以使用数据库系统监视器并借助后阈值排序（*post_threshold_sorts*）监视元素来跟踪排序活动。

## spm_log_file_sz - 同步点管理器日志文件大小

此参数以 4 KB 页为单位标识“同步点管理器”（SPM）日志文件的大小。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

256 [4 - 1000]

**计量单位**

页 (4 KB)

日志文件包含在 sqllib 下的 spmlog 子目录中，并且在首次启动 SPM 时创建。

**建议**：同步点管理器日志文件大小应适中，以便既可维护性能又防止浪费空间。所需的大小取决于使用受保护对话的事务数以及发出 COMMIT 或 ROLLBACK 的频率。

要更改 SPM 日志文件的大小:

1. 通过使用 LIST DRDA INDOUBT TRANSACTIONS 命令确定没有任何不确定事务。
2. 如果没有不确定事务，那么停止数据库管理器。
3. 用新的 SPM 日志文件大小来更新数据库管理器配置。
4. 转至 $HOME/sqllib 目录，并发出 rm -fr spmlog 命令以删除当前 SPM 日志。（注意：显示的是 AIX 命令。其他系统可能会需要不同的除去或删除命令。）
5. 启动数据库管理器。在启动数据库管理器时创建一个指定大小的新的 SPM 日志。

## spm_log_path - 同步点管理器日志文件路径

此参数指定将“同步点管理器”(SPM) 日志写入的目录。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

sqllib/spmlog [任何有效路径或设备]

缺省情况下，将这些日志写入 sqllib/spmlog 目录，这在有大量事务的环境中可导致 I/O 瓶颈。使用此参数来将 SPM 日志文件放在比当前 sqllib/spmlog 目录更快的磁盘上。它允许在 SPM 代理程序中有更好的并行性。

## spm_max_resync - 同步点管理器再同步代理程序限制

此参数标识可同时执行再同步操作的代理程序数。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器

- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

20 [10 — 256 ]

## spm_name - 同步点管理器名

此参数向数据库管理器标识“同步点管理器”（SPM）实例的名称。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值** 从 TCP/IP 主机名派生

## srvcon_auth - 用于服务器中的入局连接的认证类型

此参数指定当处理服务器上的入局连接时如何进行用户认证以及在何处进行用户认证；它用来重设当前认证类型。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

Null [ CLIENT; SERVER; SERVER_ENCRYPT; KERBEROS; KRB_SERVER_ENCRYPT; GSSPLUGIN; GSS_SERVER_ENCRYPT ]

如果未指定值，那么 DB2 使用 *authentication* 数据库管理器配置参数的值。

有关每种认证类型的描述，请参阅 第 431 页的『authentication - 认证类型』。

## srvcon_gssplugin_list - 用于服务器中的入局连接的 GSS API 插件列表

此参数指定受数据库服务器支持的 GSS API 插件库。当将 *srvcon_auth* 参数指定为 KERBEROS、KRB_SERVER_ENCRYPT、GSSPLUGIN 或 GSS_SERVER_ENCRYPT

时，或者当未指定 *srvcon_auth* 且将认证类型指定为 KERBEROS、KRB_SERVER_ENCRYPT、GSSPLUGIN 或 GSS_SERVER_ENCRYPT 时，此参数处理服务器中的入局连接。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

Null [任何有效字符串]

缺省情况下，该值为 NULL。如果认证类型为 GSSPLUGIN 且此参数为 NULL，那么会返回错误。如果认证类型为 KERBEROS 且此参数为 NULL，那么使用 DB2 提供的 Kerberos 模块或库。如果使用另一种认证类型，那么不使用此参数。

当认证类型是 KERBEROS 且此参数的值不为 NULL，那么列表必须只包含一个 Kerberos 插件并且将该插件用于认证（忽略列表中的所有其他 GSS 插件）。如果存在多个 Kerberos 插件，那么会返回一条错误。

必须用逗号（,）将每个 GSS API 插件名称隔开，并且逗号前后不能有空格。应该按首选项的顺序来列示插件名称。

## srvcon_pw_plugin - 用于服务器中的入局连接的用户标识密码插件

此参数指定要用于服务器端认证的缺省用户标识密码插件库的名称。当将 *srvcon_auth* 参数指定为 CLIENT、SERVER、SERVER_ENCRYPT 或 DATA_ENCRYPT 时，或者当未指定 *srvcon_auth* 且将 *authentication* 指定为 CLIENT、SERVER、SERVER_ENCRYPT 或 DATA_ENCRYPT 时，此参数还处理服务器中的入局连接。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

Null [任何有效字符串]

缺省情况下，值为 NULL 并且使用 DB2 提供的用户标识密码插件库。该插件将用于所有组查找。对于非 root 用户安装，如果使用了 DB2 用户标识和密码插件库，那么 db2rfe 命令必须在运行才能使用 DB2 产品。

## srv_plugin_mode - 服务器插件方式

此参数指定是以设防方式还是以不设防方式运行插件。不设防方式是唯一受支持的方式。

**配置类型**
数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**
可配置

**缺省值 [范围]**
UNFENCED

## start_stop_time - 启动和停止超时

此参数以分钟为单位指定时间，在该段时间内所有数据库分区服务器必须响应 DB2START 或 DB2STOP 命令。在 ADD DBPARTITIONNUM 操作期间，它也用作超时值。

**配置类型**
数据库管理器

**适用于** 带有本地和远程客户机的数据库服务器

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
10 [1 - 1 440]

**计量单位**
分钟

在指定的时间内未响应 DB2START 命令的数据库分区服务器向此实例的主目录下 sqllib 子目录的 log 子目录中的 db2start 错误日志发送一条消息。应在重新启动这些节点之前对这些节点发出 DB2STOP 命令。

在指定的时间内未响应 DB2STOP 命令的数据库分区服务器向此实例的主目录下 sqllib 子目录的 log 子目录中的 db2stop 错误日志发送一条消息。您可以对每个未响应的数据库分区服务器或者对所有服务器发出 db2stop。（那些已停止的服务器将返回一条信息以表明它们是停止的。）

如果多分区数据库中的 db2start 或 db2stop 操作未在 *start_stop_time* 数据库管理器配置参数所指定的值内完成，那么超时的数据库分区将内部终止。如果具有许多数据库分区的环境的 *start_stop_time* 值较低，那么可能会遇到此行为。要解决这种情况，增大 *start_stop_time* 的值。

在使用 DB2START、START DATABASE MANAGER 或 ADD DBPARTITIONNUM 命令之一添加新数据库分区时，添加数据库分区操作必须确定实例中的每个数据库是否已启用自动存储器。这是通过与每个数据库的目录分区通信完成的。如果已启用自动存储器，就会在该通信过程中检索存储路径定义。同样，如果要创建带有数据库分区的系统临时表空间，该操作就可能必须与另一数据库分区服务器通信以检索该服务器上数据库分区的表空间定义。在确定 *start_stop_time* 参数值时，应该考虑这些因素。

## svcename - TCP/IP 服务名称

此参数包含数据库服务器将用于等待来自远程客户机节点的通信的 TCP/IP 端口的名称。此名称必须保留给数据库管理器使用。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值** Null

为了使用 TCP/IP 接受来自数据服务器运行时客户机的连接请求，数据库服务器必须侦听指定给该服务器的端口。数据库服务器的系统管理员必须保留一个端口（端口号为 *n*）并在服务器的 services 文件中定义其相关 TCP/IP 服务名称。

需要在数据库客户机上的 services 文件中定义数据库服务器端口（端口号为 *n*）和它的 TCP/IP 服务名称。

在 Linux 和 UNIX 系统上，服务文件位于 `/etc/services` 中。

应将 *svcename* 参数设置为与主连接端口相关联的端口号或服务名称，以便启动数据库服务器时，它可以确定在哪个端口上侦听入局连接请求。

## sysadm_group - 系统管理权限组名

此参数定义具有数据库管理器实例的 SYSADM 权限的组名。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值** Null

系统管理（SYSADM）权限是数据库管理器中最高级别的权限，它控制所有数据库对象。

SYSADM 权限是由用于特定的操作环境中的安全性工具确定的。

- 对于 Windows 操作系统，可将此参数设置为在 Windows 安全性数据库中定义的任何本地组。组名的长度必须符合在 SQL 和 XML 限制中指定的长度限制。如果对此参数指定“NULL”，那么 Administrators 组的所有成员都具有 SYSADM 权限。
- 对于 Linux 和 UNIX 系统，如果指定“NULL”作为此参数的值，那么 SYSADM 组缺省为实例所有者的主组。

  如果该值不是“NULL”，那么 SYSADM 组可以是任何有效的 UNIX 组名。

要将该参数复原为其缺省值（NULL），可使用 UPDATE DBM CFG USING SYSADM_GROUP NULL。必须用大写字母指定关键字“NULL”。

## sysctrl_group - 系统控制权限组名

此参数定义具有系统控制（SYSCTRL）权限的组名。SYSCTRL 具有一些特权，这些特权允许执行影响系统资源的操作但不允许直接访问数据。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值** Null

在所有平台上，只要组名的长度符合在 SQL 和 XML 限制中指定的长度限制，它们就是可接受的。

## sysmaint_group - 系统维护权限组名

此参数定义具有系统维护（SYSMAINT）权限的组名。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值** Null

SYSMAINT 具有对所有与实例相关的数据库执行维护操作的特权，但没有对数据的直接访问权。

在所有平台上，只要组名的长度符合在 SQL 和 XML 限制中指定的长度限制，它们就是可接受的。

**注意**：对于 Windows 客户机，在使用系统安全性时（即 **authentication** 是 CLIENT、SERVER 或任何其他有效认证），此参数必须为 NULL。这是因为 Windows 操作系统不存储组信息，因此不提供任何方法来确定一个用户是否是指定的 SYSMAINT 组的成员。指定组名时，所有用户都不能是它的成员。

要将该参数复原至它的缺省值（NULL），请使用 UPDATE DBM CFG USING SYSMAINT_GROUP NULL。必须用大写字母指定关键字 NULL。

## sysmon_group - 系统监视权限组名

此参数定义具有系统监视（SYSMON）权限的组名。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值** Null

在实例级具有 SYSMON 权限的用户能够为数据库管理器实例或它的数据库生成数据库系统监视器快照。SYSMON 权限能够使用下列命令:

- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES
- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

具有 SYSADM、SYSCTRL 或 SYSMAINT 权限的用户自动具有生成数据库系统监视器快照和使用这些命令的能力。

在所有平台上，只要组名的长度符合在 SQL 和 XML 限制中指定的长度限制，它们就是可接受的。

要将该参数复原至它的缺省值（NULL），请使用 UPDATE DBM CFG USING SYSMON_GROUP NULL。必须用大写字母指定关键字 NULL。

## tm_database - 事务管理器数据库名称

此参数标识每个 DB2 实例的“事务管理器”（TM）数据库的名称。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

1ST_CONN [任何有效数据库名称]

TM 数据库可以是:

- 本地 DB2 数据库
- 不位于主机或 AS/400 系统上的远程 DB2 数据库
- DB2 OS/390 版版本 5 数据库（如果通过 TCP/IP 访问且未使用同步点管理器（SPM））。

TM 数据库是用作记录器和协调程序的数据库，并用来对不确定事务执行恢复。

可以将此参数设置为 **1ST_CONN**，这样将把 TM 数据库设置为用户所连接的第一个数据库。

**建议**：为了简化管理和操作，您可能希望对许多实例创建几个数据库，并将这些数据库专门用作 TM 数据库。

## tp_mon_name - 事务处理器监视器名

此参数标识正在使用的事务处理（TP）监视器产品的名称。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值**

无缺省值

**有效值**

- CICS®
- MQ
- CB
- SF
- TUXEDO
- TOPEND
- 空白或其他值（对于 UNIX 和 Windows；对于 Solaris 或 SINIX 则没有其他可能值）

- 如果应用程序在 WebSphere Enterprise Server Edition CICS 环境中运行，那么此参数应设置为“CICS”
- 如果应用程序在 WebSphere Enterprise Server Edition Component Broker 环境中运行，那么此参数应设置为“CB”
- 如果应用程序在 IBM MQSeries® 环境中运行，那么此参数应设置为“MQ”
- 如果应用程序在 BEA Tuxedo 环境中运行，那么此参数应设置为“TUXEDO”
- 如果应用程序在 IBM San Francisco 环境中运行，那么此参数应设置为“SF”。

**IBM WebSphere EJB** 和 **Microsoft Transaction Server** 用户不需要对此参数配置任何值。

如果不是使用上述产品，那么不应配置此参数，而应保留它为空白。

在 Windows 上的先前版本的 IBM DB2 中，此参数包含 DLL 的路径和名称，而该 DLL 包含“XA 事务管理器”的函数 *ax_reg* 和 *ax_unreg*。此格式仍然受支持。如果此参数的值与上述任何“TP 监视器”名不匹配，那么将假设它的值是包含 *ax_reg* 和 *ax_unreg* 函数的库名。对于 UNIX 和 Windows 环境就是这样的。

**TXSeries CICS 用户**: 在 Windows 上的此产品的先前版本中，需要将此参数配置为“libEncServer:C”或“libEncServer:E”。虽然此配置仍受支持，但不再要求这样。将此参数配置成“CICS”就可以了。

**MQSeries 用户**: 在 Windows 上此产品的先前版本中，需要将此参数配置为“mqmax”。虽然此配置仍受支持，但不再要求这样。将此参数配置成“MQ”就可以了。

**Component Broker 用户**: 在 Windows 上此产品的先前版本中，需要将此参数配置为“somtrx1i”。虽然此配置仍受支持，但不再要求这样。将此参数配置成“CB”就可以了。

**San Francisco 用户**: 在 Windows 上此产品的先前版本中，需要将此参数配置为“ibmsfDB2”。虽然此配置仍受支持，但不再要求这样。将此参数配置成“SF”就可以了。

可为此参数指定的字符串的最大长度为 19 个字符。

也有可能在 IBM DB2 版本 9.1 的 XA OPEN 字符串中配置此信息。如果多个 “事务处理监视器”正在使用单一 DB2 实例，那么必须使用此功能。

## trust_allclnts - 信赖所有客户机

此参数和 *trust_clntauth* 用来确定在何处向数据库环境验证用户。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

YES [NO, YES, DRDAONLY]

仅当将 *authentication* 参数设置为 CLIENT 时，此参数才是活动的。

通过接受此参数的缺省值“YES”，所有客户机都作为可信的客户机对待。这意味着，服务器假定在客户机中安全性级别可用并且可在客户机上验证用户。

仅当 *authentication* 参数设置为 CLIENT 时，才能将此参数更改为“NO”。如果此参数设置为“NO”，那么不可信客户机在连接到服务器时必须提供用户标识和密码组合。不可信客户机为没有用于认证用户的安全子系统的操作系统平台。

将此参数设置为“DRDAONLY”可防止认证来自除 DB2 OS/390 和 z/OS 版、DB2 VM 和 VSE 版以及 DB2 OS/400® 版的客户机之外的所有客户机。只有这些客户机可信赖，才能执行客户端的认证。所有其他客户机必须提供用户标识和密码，以供服务器认证。

将 *trust_allclnts* 设置为“DRDAONLY”时，使用 *trust_clntauth* 参数来确定在何处认证客户机。如果将 *trust_clntauth* 设置为“CLIENT”，那么在客户机中进行认证。如果将 *trust_clntauth* 设置为“SERVER”，那么在未提供密码时在客户机中进行认证，而在提供了密码时在服务器中进行认证。

## trust_clntauth - 可信客户机认证

此参数指定，当客户机提供用于连接的用户标识和密码组合时，是在服务器还是在客户机中认证可信的客户机。仅当将 *authentication* 参数设置为 CLIENT 时，此参数（和 *trust_allclnts*）才是活动的。如果不提供用户标识和密码，那么假定客户机已验证该用户，因而在服务器上不需要进一步验证。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**
CLIENT [CLIENT, SERVER]

如果此参数设置为 CLIENT（缺省值），那么无须提供用户标识和密码组合，可信客户机就能连接，并假定操作系统已认证该用户。如果此参数设置为 SERVER，那么将在服务器上验证用户标识和密码。

CLIENT 的数值是 0。SERVER 的数值是 1。

## util_impact_lim - 实例影响策略

此参数允许数据库管理员（DBA）限制已调速实用程序对工作负载的性能降级。

**配置类型**
数据库管理器

**适用于**

- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
10 [1 - 100 ]

**计量单位**
允许对工作负载产生的影响的百分比

如果性能降级受限制，那么 DBA 可在关键生产时间段运行联机实用程序，并保证对生产工作的性能影响在可接受的限制之内。

例如，指定 *util_impact_lim*（影响策略）值为 10 的 DBA 可能期望已调速备份调用影响的工作负载不会超过 10%。

如果 *util_impact_lim* 为 100，那么不会对任何实用程序调用进行调速。在这种情况下，实用程序可能对工作负载有任意的（和不期望的）影响。如果将 *util_impact_lim* 设置为小于 100 的值，那么可以调速方式调用实用程序。要以调速方式运行实用程序，还必须使用非零优先级调用该实用程序。

**建议**：大多数用户将从将 *util_impact_lim* 设置为低值（例如，1 与 10 之间的值）中受益。

完成已调速实用程序所花的时间通常比完成非调速实用程序要长。如果发现实用程序运行的时间过长，那么增大 *util_impact_lim* 的值，或通过将 *util_impact_lim* 设置为 100 来整体禁用调速。

# 数据库配置参数

## alt_collate - 备用整理顺序

此参数指定了要用于非 Unicode 数据库中的 Unicode 表的整理顺序。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

Null [IDENTITY_16BIT ]

只有设置了此参数之后，才能在非 Unicode 数据库中创建 Unicode 表和例程。一旦设置了此参数，就不能更改或将它复位。

不能对 Unicode 数据库设置此参数。

## app_ctl_heap_sz - 应用程序控制堆大小

在版本 9.5 中不推荐使用此参数，但是版本 9.5 之前的数据服务器和客户机仍然使用此参数。DB2 V9.5 数据库管理器将忽略对此配置参数指定的任何值。在 V9.5 中，它已被 *appl_memory* 配置参数取代。

**注:** 下列信息仅适用于版本 9.5 之前的数据服务器和客户机。

对于分区数据库以及启用了内部并行性 (intra_parallel=ON) 的非分区数据库，此参数指定分配给应用程序的共享内存区的平均大小。对于禁用内部并行性的非分区数据库 (intra_parallel=OFF)，这是将为堆分配的最大专用内存。每个数据库分区的每个连接有一个应用程序控制堆。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

**带有本地和远程客户机的数据库服务器**

- 未启用 INTRA_PARALLEL 时: 128 [1 - 64 000]
- 启用了 INTRA_PARALLEL 时: 512 [1 - 64 000]

**带有本地客户机的数据库服务器**

- 未启用 INTRA_PARALLEL 时: 64 [1 - 64 000]（适用于非 UNIX 平台）

- 启用了 INTRA_PARALLEL 时：512 [1 - 64 000]（适用于非 UNIX 平台）
- 未启用 INTRA_PARALLEL 时：128 [1 - 64 000]（适用于 Linux 和 UNIX 平台）
- 启用了 INTRA_PARALLEL 时：512 [1 - 64 000]（适用于 Linux 和 UNIX 平台）

**带有本地和远程客户机的分区数据库服务器**
512 [1 - 64 000]

**计量单位**
页 (4 KB)

**分配时间**
当应用程序启动时

**释放时间**
当应用程序完成时

主要是在代表同一请求工作的代理程序之间共享信息时需要此应用程序控制堆。对于非分区数据库，当运行带有并行度等于 1 的查询时，此堆所使用的资源最少。

这个堆还用来存储已声明临时表的描述符信息。所有尚未显式删除的已声明临时表的描述符信息存放在这个堆的内存中，在删除已声明临时表之前，不能删除这些信息。

**建议**：最初，从缺省值开始。如果在运行复杂的应用程序或您的系统包含大量的数据库分区，又或者您使用已声明临时表，那么可能需要将该值设置得高一些。所需的内存量随并发活动的已声明临时表数的增加而增加。与带有较少列的表相比，带有许多列的已声明临时表的表描述符大小要大，因此，如果应用程序的已声明临时表中有相当多的列，那么也会增加对应用程序控制堆的需求。

## appgroup_mem_sz - 应用程序组内存集的最大大小

在版本 9.5 中不推荐使用此参数，但是版本 9.5 之前的数据服务器和客户机仍然使用此参数。DB2 V9.5 数据库管理器将忽略对此配置参数指定的任何值。在 V9.5 中，它已被 *appl_memory* 配置参数取代。

**注**：下列信息仅适用于版本 9.5 之前的数据服务器和客户机。

此参数确定应用程序组共享内存段的大小。

**配置类型**
数据库

**参数类型**
可配置

**缺省值 [范围]**

**带有本地客户机的 UNIX 数据库服务器（32 位 HP-UX 除外）**
20 000 [1 - 1 000 000 ]

**32 位 HP-UX**

- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的数据库服务器

- 带有本地和远程客户机的分区数据库服务器

10 000 [1 - 1 000 000 ]

**带有本地客户机的 Windows 数据库服务器**
10 000 [1 - 1 000 000 ]

**带有本地客户机和远程客户机的数据库服务器（32 位 HP-UX 除外）**
30 000 [1 - 1 000 000 ]

**带有本地客户机和远程客户机的分区数据库服务器（32 位 HP-UX 除外）**
40 000 [1 - 1 000 000 ]

**计量单位**
页 (4 KB)

需要在工作在同一应用程序上的代理程序之间共享的信息存储在应用程序组共享内存段中。

在分区数据库中，或在启用了分区内并行性或集中器的非分区数据库中，多个应用程序共享一个应用程序组。为应用程序组分配了一个应用程序组共享内存段。在应用程序组共享内存段中，每个应用程序将有它自己的应用程序控制堆，并且所有应用程序将共享一个应用程序组共享堆。

一个应用程序组中的应用程序的数目通过以下公式计算:

```
appgroup_mem_sz / app_ctl_heap_sz
```

应用程序组共享堆大小通过以下公式计算:

```
appgroup_mem_sz * groupheap_ratio / 100
```

每个应用程序控制堆的大小通过以下公式计算:

```
app_ctl_heap_sz * (100 - groupheap_ratio) / 100
```

**建议**: 除非遇到性能问题，否则请保留此参数的缺省值。

## appl_memory - 应用程序内存配置参数

此参数允许 DBA 和 ISV 控制 DB2 数据库代理程序分配的用于为应用程序请求提供服务的最大应用程序内存量。缺省情况下，此参数的值设置为 AUTOMATIC，这表示只要数据库分区分配的总内存量未超过 *instance_memory* 限制，就允许所有应用程序内存请求。

**配置类型**
数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**
可联机配置

**缺省值 [范围]**
Automatic [128 - 4 294 967 295]

**计量单位**

页 (4 KB)

**分配时间**

在数据库激活期间

**释放时间**

在数据库取消激活期间

**注:** *appl_memory* 设置为 AUTOMATIC 时，在数据库激活期间分配的初始应用程序内存最小，并且会根据需要增大（或减小）。将在内存中应用更改，并且 *appl_memory* 的值在磁盘上不会更改，与 db2 get db cfg show detail 所显示的结果相同。在下一次激活时将重新计算此值。如果 *appl_memory* 设置为特定值，那么在数据库激活期间最初会分配请求的内存量，并且应用程序内存大小不会改变。如果不能从操作系统分配初始应用程序内存量，或初始应用程序内存量超过 *instance_memory* 限制，那么数据库激活将失败，并且出现 SQL1084C 错误（不能分配共享内存段）。

## applheapsz - 应用程序堆大小

在先前发行版中，*applheapsz* 数据库配置参数指的是为应用程序工作的每个单独数据库代理程序可以消耗的应用程序内存量。对于版本 9.5，*applheapsz* 指的是整个应用程序可以消耗的应用程序内存的总量。对于 DPF、集中器或 SMP 配置，这表示在类似工作负载下，除非使用了 AUTOMATIC 设置，否则可能需要增大在先前发行版中使用的 *applheapsz* 值。

对于版本 9.5，此数据库配置参数的缺省值为 AUTOMATIC，这表示它将根据需要增大，直到达到 *appl_memory* 限制或达到 *instance_memory* 限制。

**配置类型**

数据库

**参数类型**

可联机配置

**缺省值 [范围]**

Automatic [16 - 60 000]

**计量单位**

页 (4 KB)

**分配时间**

当应用程序与数据库关联或连接至数据库时。

**释放时间**

当应用程序取消与数据库的关联或断开与数据库的连接时。

**注:** 此参数定义最大应用程序堆大小。当应用程序第一次与数据库连接时，将为每个数据库应用程序分配一个应用程序堆。该堆将由为该应用程序工作的所有数据库代理程序共享。（在先前发行版中，每个数据库代理程序都分配它自己的应用程序堆。）将根据需要从应用程序堆中分配用于处理应用程序的内存，最大为此参数指定的限制。将参数设置为 AUTOMATIC 时，允许应用程序堆根据需要增长，直到到达数据库的 *appl_memory* 限制或数据库分区的 *instance_memory* 限制。当应用程序断开与数据库的连接时，将释放整个应用程序堆。

联机更改的值在应用程序连接边界处生效，这也就是说，在动态更改值后，当前连接的应用程序仍使用旧值，但所有新近连接的应用程序将使用新值。

## archretrydelay - 出错时的归档重试延迟

此参数指定在对日志文件尝试进行归档失败之后再次尝试归档之前要等待的秒数。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

20 [0 - 65 535 ]

仅当 *numarchretry* 数据库配置参数的值至少为 1 时，后续的重试才会生效。

## auto_del_rec_obj - 自动删除恢复对象配置参数

此参数指定在修剪数据库日志文件、备份映像和装入副本映像的关联恢复历史记录文件条目时，是否应删除这些对象。

**配置类型**

数据库

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

OFF [ ON; OFF]

可以使用 PRUNE HISTORY 命令或 db2Prune API 修剪恢复历史记录文件中的条目。还可以配置 IBM 数据服务器数据库管理器，以便在每次完全备份数据库后自动修剪恢复历史记录文件。如果将 *auto_del_rec_obj* 数据库配置参数设置为 ON，那么数据库管理器在修剪历史记录文件时还将删除相应的物理日志文件、备份映像和装入副本映像。仅当存储介质为磁盘时，或者您在使用 Tivoli Storage Manager 之类的存储管理器时，数据库管理器才能删除数据库日志、备份映像和装入副本映像。

## auto_maint - 自动维护

此参数是所有其他自动维护数据库配置参数（*auto_db_backup*、*auto_tbl_maint*、*auto_runstats*、*auto_stats_prof*、*auto_stmt_stats*、*auto_prof_upd* 和 *auto_reorg*）的父参数。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
ON [ ON; OFF]

当禁用此参数时，也就禁用了它的所有子参数，但是，数据库配置文件中所记录的这些子参数的设置不会更改。当启用此父参数时，记录的它的子参数的值就会生效。因此，可以全局启用或禁用自动维护。

缺省情况下，此参数设置为 ON。

通过设置下列参数可以单独地启用或禁用各个自动维护功能:

**auto_db_backup**
此自动维护参数将启用或禁用数据库的自动备份操作。备份策略（已定义的一组规则或准则）可以用来指定自动行为。备份策略的目标是确保定期备份数据库。当第一次运行“DB2 运行状况监视器”时，会自动创建数据库的备份策略。缺省情况下，此参数设置为 OFF。要启用此参数，必须将它设置为 ON，而且还必须启用其父参数。

**auto_tbl_maint**
此参数是所有表维护参数（*auto_runstats*、*auto_stats_prof*、*auto_prof_upd* 和 *auto_reorg*）的父参数。当禁用此参数时，也就禁用了它的所有子参数，但是，数据库配置文件中所记录的这些子参数的设置不会更改。当启用此父参数时，记录的它的子参数的值就会生效。因此，可以全局启用或禁用表维护。

缺省情况下，此参数设置为 ON。

**auto_runstats**
此自动维护表参数将启用或禁用数据库的自动表运行统计操作。运行统计策略（已定义的一组规则或准则）可以用来指定自动行为。优化器使用 runstats 实用程序收集的统计信息来确定访问物理数据的最佳方案。如果要启用，那么必须将此参数设置为 ON，并且还必须启用它的父参数。

缺省情况下，此参数设置为 ON。

**auto_stats_prof**
当启用此参数时，此自动维护表参数将打开统计概要文件的生成，生成统计概要文件可以用来提高这样一些应用程序的性能：它们的工作负载包括对几个表的复杂查询、许多谓词、连接和分组操作。要启用此参数，必须将它设置为 ON，而且还必须启用其父参数。

缺省情况下，此参数设置为 OFF。

**auto_stmt_stats**

此参数启用和禁用收集实时统计信息。它是 *auto_runstats* 配置参数的子代。仅当父 *auto_runstats* 配置参数也启用时，才启用此功能。例如，要启用

*auto_stmt_stats*，将 *auto_maint*、*auto_tbl_maint* 和 *auto_runstats* 设置为 ON。要保留子代值，*auto_runstats* 配置参数可以为 ON，而 *auto_maint* 配置参数为 OFF。相应的“自动 Runstats”功能仍将为 OFF。

假定同时启用了“自动 Runstats”和“自动重组”功能，那么设置将如下所示:

```
自动维护                          (AUTO_MAINT) = ON
  自动数据库备份                 (AUTO_DB_BACKUP) = OFF
    自动表维护                   (AUTO_TBL_MAINT) = ON
      自动 runstats              (AUTO_RUNSTATS) = ON
        自动语句统计信息    (AUTO_STMT_STATS) = OFF
      自动进行统计信息概要分析     (AUTO_STATS_PROF) = OFF
        自动概要文件更新           (AUTO_PROF_UPD) = OFF
        自动重组                   (AUTO_REORG) = ON
```

通过将 *auto_tbl_maint* 设置为 OFF，可以暂时禁用“自动 Runstats”和“自动重组”功能。以后可以通过将 *auto_tbl_maint* 重新设置为 ON 来启用这两个功能。不需要发出 db2stop 或 db2start 命令来使更改生效。

缺省情况下，此参数设置为 OFF。

**auto_prof_upd**

当启用此参数时，此自动维护表参数（*auto_stats_prof* 的子参数）指定要使用建议来更新 runstats 概要文件。当禁用此参数时，会将建议存储在 opt_feedback_ranking 表中，在手动更新 runstats 概要文件时可检查该表。要启用此参数，必须将它设置为 ON，而且还必须启用其父参数。

缺省情况下，此参数设置为 OFF。

**auto_reorg**

此自动维护表参数将启用或禁用数据库的自动重组表和索引。重组策略（已定义的一组规则或准则）可以用来指定自动行为。要启用此参数，必须将它设置为 ON，而且还必须启用其父参数。

缺省情况下，此参数设置为 OFF。

## autorestart - 允许自动重新启动

此参数确定在应用程序连接至数据库时，如果该数据库异常终止，数据库管理器是否可以自动调用重新启动数据库实用程序。

**配置类型**

数据库

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

On [ On; Off ]

如果在应用程序连接至数据库时该数据库异常终止（例如，由于电源故障或系统软件故障），那么重新启动数据库实用程序将执行*崩溃恢复*。崩溃恢复应用发生故障时在数据库缓冲池中但未写入磁盘的任何已落实的事务。崩溃恢复还会取消可能已经写入磁盘的任何未落实的事务。

如果未启用 *autorestart*，那么试图与需要执行崩溃恢复（需要重新启动）的数据库连接的应用程序将接收到 SQL1015N 错误。在这种情况下，该应用程序可调用重新启动数据库实用程序，或者您可以通过选择该恢复工具的重新启动操作来重新启动数据库。

## avg_appls - 平均活动应用程序数

查询优化器使用此参数来帮助估计在运行时将有多少缓冲池可用于选择的访问方案。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 语句边界

**缺省值 [范围]**
Automatic [1 – maxappls ]

**计量单位**
计数器

**建议**: 当在多用户环境（特别是使用复杂查询和大缓冲池的环境中）运行 DB2 时，您可能希望查询优化器知道多个查询用户正在使用您的系统，以便该优化器应该在判断缓冲池的可用性时更保守。

在设置此参数时，应估计通常使用数据库的复杂查询应用程序的数目。此估计应排除所有小型的 OLTP 应用程序。如果对此数求值有困难，可以乘以下列数:

- 对数据库运行的所有应用程序的平均数。数据库系统监视器可提供关于任何给定时间的应用程序数的信息，从而可以使用采样技术来计算一段时间内的平均值。来自数据库系统监视器的信息包括 OLTP 和非 OLTP 应用程序。
- 对于复杂查询应用程序的百分比的估计。

当 avg_appls 设置为 **AUTOMATIC** 时，将在激活数据库时确定实用程序的值。

正如调整影响优化器的其他配置参数一样，应以较小的增量调整此参数。这样使您能够将路径选择差别减至最小。

在更改此参数之后应考虑重新绑定应用程序（使用 REBIND PACKAGE 命令）。

## backup_pending - 备份暂挂指示器

此参数指示在访问数据库之前是否需要对该数据库进行完全备份。

**配置类型**
数据库

**参数类型**
参考

此参数仅在下列情况下才为 on: 数据库配置已更改，以便该数据库从不可恢复转变为可恢复（即，最初 *logretain* 和 *userexit* 参数都设置为 NO，然后将这两个参数中的一个或两个设置为 YES，并接受对该数据库配置的更新。）

## blk_log_dsk_ful - 磁盘已满时阻止进行日志记录

可以设置此参数以防止当 DB2 不能在活动日志路径中创建新日志文件时出现“磁盘已满”错误。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
No [ Yes; No ]

DB2 将尝试每隔五分钟就创建一次日志文件，直至成功，而不是生成“磁盘已满”错误。在每次尝试之后，DB2 会将消息写入“管理通知”日志。监视“管理通知”日志是确认应用程序是否由于日志磁盘已满而挂起的唯一方法。在成功创建日志文件之前，尝试更新表数据的所有用户应用程序都不能落实事务。只读查询可能不会直接受影响；但是，如果查询需要访问被更新请求锁定的数据或者由更新应用程序在缓冲池中修正的数据页时，只读查询也将像挂起一样。

将 *blk_log_dsk_ful* 设置为 *yes* 会使 DB2 遇到日志磁盘已满错误时应用程序挂起，从而使您可解决该错误并允许事务完成。可以通过将旧日志文件移至另一个文件系统或扩充文件系统来解决磁盘已满情况，以便挂起的应用程序可以完成。

如果 *blk_log_dsk_ful* 设置为 *no*，那么接收日志磁盘已满错误的事务将会失败并回滚。在某些情况下，如果事务导致日志磁盘已满错误，数据库将关闭。

## blocknonlogged -“禁止创建允许不进行日志记录的活动的表”

此参数指定数据库管理器是否将允许表激活 NOT LOGGED 或 NOT LOGGED INITIALLY 属性。

**配置类型**
数据库

**参数类型**
可联机配置

**缺省值 [范围]**
No [Yes, No]

DB2 版本 9.5 修订包 4 和更高版本中提供了此数据库配置参数。

### 使用说明

- 如果 **blocknonlogged** 设置为 YES，那么当下列任一情况存在时，CREATE TABLE 和 ALTER TABLE 语句将失败:
  - 指定了 NOT LOGGED INITIALLY 参数。
  - 对 LOB 列指定了 NOT LOGGED 参数。
- 如果 **blocknonlogged** 设置为 YES，那么当下列情况存在时，LOAD 命令将失败:
  - 您指定了 NONRECOVERABLE 选项。
  - 您指定了 COPY NO 选项。

## catalogcache_sz - 目录高速缓存大小

此参数指定目录高速缓存可以使用的数据库堆中的最大空间（以页计）。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
-1 [MAXAPPLS*5]

**计量单位**
页 (4 KB)

**分配时间**
当初始化数据库时

**释放时间**
当数据库关闭时

此参数是在数据库共享内存外分配的，并且用于高速缓存系统目录信息。在分区数据库系统中，每个数据库分区都有一个目录高速缓存。

高速缓存各个数据库分区中的目录信息允许数据库管理器不需要访问系统目录（或分区数据库环境中的目录节点）来获取先前检测的信息，从而降低其内部开销。使用目录高速缓存可以帮助提高下列各项的整体性能:

- 绑定程序包以及编译 SQL 和 XQuery 语句
- 涉及到检查数据库级别特权、例程特权、全局变量特权和角色权限的操作
- 连接至分区数据库环境中的非目录节点的应用程序

通过采用服务器或分区数据库环境中的缺省值（-1），用来计算页分配的值是为 *maxappls* 配置参数指定的值的五倍。如果 *maxappls* 的五倍小于 8，那么会出现例外情况。在此情况下，缺省值 -1 会将 *catalogcache_sz* 设置为 8。

**建议**: 开始使用缺省值，并使用数据库系统监视器来进行调整。当调整此参数时，应考虑如果为目录高速缓存保留的额外内存是为另一目的分配的（如缓冲池或程序包高速缓存），它是否会更有效。

如果工作负载涉及到在短时间内编译许多 SQL 或 XQuery，且其后很少或不进行编译，那么调整此参数尤其重要。如果高速缓存太大，那么可能会因保留不再使用的信息的副本而浪费内存。

在分区数据库环境中，考虑是否需要将目录节点上的 *catalogcache_sz* 设置为更大的值，因为非目录节点上要求的目录信息将始终首先在目录节点上高速缓存。

*cat_cache_lookups*（目录高速缓存查询）、*cat_cache_inserts*（目录高速缓存插入）、*cat_cache_overflows*（目录高速缓存溢出）和 *cat_cache_size_top*（目录高速缓存高水位标记）监视元素将帮助您确定是否应调整此配置参数。

**注**: 目录高速缓存在分区数据库环境中的所有节点上存在。因为每个节点都有本地数据库配置文件，所以每个节点的 *catalogcache_sz* 值定义本地目录高速缓存的大小。为了

提供足够的高速缓存并避免溢出情况的发生，需要在每个节点上显式设置 *catalogcache_sz* 值，并考虑将非目录节点上的 *catalogcache_sz* 设置为比在目录节点上的该值小的可能性；记住将从目录节点的高速缓存中检索需要在非目录节点高速缓存的信息。因此，在非目录节点的目录高速缓存就像目录节点上的目录高速缓存中的信息的子集。

通常情况下，如果工作单元包含几个动态 SQL 或 XQuery 语句或如果正在绑定包含许多静态 SQL 或 XQuery 语句的程序包，那么需要较多的高速缓存空间。

## chngpgs_thresh - 更改的页数阈值

此参数指定已更改的页的级别（百分比），如果异步页清除程序当前处于不活动状态，那么将从该级别启动这些程序。

**配置类型**
数据库

**参数类型**
可配置

**缺省值 [范围]**
60 [5 – 99 ]

**计量单位**
百分比

异步页清除程序在数据库代理程序需要缓冲池中的空间之前将更改的页从缓冲池中写入磁盘。因此，数据库代理程序应该不必等待写出已更改的页，它们也能使用缓冲池中的空间。这提高了数据库应用程序的整体性能。

当启动页清除程序时，页清除程序将构建要写入磁盘的页的列表。一旦页清除程序已将这些页写入磁盘，那么将再次变为不活动的，并等待下一个触发器来启动。

当设置了 DB2_USE_ALTERNATE_PAGE_CLEANING 注册表变量时（即，使用了进行页清理的另一种方法），*chngpgs_thresh* 参数将不起作用，数据库管理器自动确定在缓冲池中要维护多少脏页。

**建议**：对于具有繁重的更新事务工作负载的数据库，您通常可以将该参数值设置为等于或小于缺省值，以确保在缓冲池中有足够的干净页。如果数据库有少量很大的表，那么大于缺省值的百分比有助于提高性能。

## codepage - 用于数据库的代码页

此参数显示创建数据库所使用的代码页。*codepage* 参数是根据 *codeset* 参数派生的。

**配置类型**
数据库

**参数类型**
参考

## codeset - 用于数据库的代码集

此参数显示创建数据库所使用的代码集。数据库管理器使用代码集来确定 *codepage* 参数值。

**配置类型**

数据库

**参数类型**

参考

## collate_info - 整理信息

此参数确定数据库的整理顺序。对于语言感知整理，前 256 个字节包含整理名的字符串表示（例如，“SYSTEM_819_US”）。

只能使用 db2CfgGet API 来显示此参数。它**不能**通过命令行处理器或控制中心来显示。

**配置类型**

数据库

**参数类型**

参考

此参数提供 260 个字节的数据库整理信息。前面 256 个字节指定数据库整理顺序，其中字节“n”包含代码点的排序权重，该代码点在数据库代码页中的基本十进制表示为“n”。

最后 4 个字节包含关于整理顺序类型的内部信息。此参数的最后四个字节是整数。该整数依据平台的字节存储次序。可能的值包括:

- **0** - 顺序包含非唯一的权重。
- **1** - 顺序包含所有唯一的权重。
- **2** - 顺序是等同顺序，对于这种顺序将逐个字节地比较字符串。
- **3** - 顺序为 NLSCHAR，用于对 TIS620-1（代码页 874）泰国语数据库中的字符进行排序。
- **4** - 顺序为 IDENTITY_16BIT，它实施“CESU-8 Compatibility Encoding Scheme for UTF–16: 8–bit”算法，该算法是在 Unicode Technical Consortium Web 站点（网址为 http: www.unicode.org）上的 Unicode Technical Report #26 中指定的。
- **X’8001’** - 顺序为 UCA400_NO，它实施基于 Unicode 标准版本 4.00 的 Unicode 整理算法（UCA），并且规范化隐式为 ON。
- **X’8002’** - 顺序为 UCA400_LTH，它实施基于 Unicode 标准版本 4.00 的 Unicode 整理算法（UCA），并按皇家泰国语字典顺序对所有泰国语字符排序。
- **X’8003’** - 顺序为 UCA400_LSK，它实施基于 Unicode 标准版本 4.00 的 Unicode 整理算法（UCA），并对所有斯洛伐克语字符作适当排序。

**注:** 对于语言感知整理，前 256 个字节包含整理名的字符串表示。

如果使用此内部类型信息，在不同平台上检索数据库的信息时需要考虑字节逆转。

可在创建数据库时指定整理顺序。

## country/region - 数据库地域代码

此参数显示用来创建数据库的*地域*代码。

**配置类型**

数据库

**参数类型**

参考

## database_consistent - 数据库处于一致状态

此参数指示数据库是否处于一致状态。

**配置类型**

数据库

**参数类型**

参考

**YES** 指示所有事务都已落实或回滚，以保证数据是一致的。如果系统在数据库一致时“崩溃”，那么不必进行任何特别操作以使该数据库可用。

**NO** 指示一个事务暂挂，或对该数据库执行的某个其他任务暂挂，且此时数据不一致。如果系统在数据库不一致时“崩溃”，那么需要使用 RESTART DATABASE 命令重新启动数据库以使该数据库可用。

## database_level - 数据库发行版级别

此参数指示可使用数据库的数据库管理器的发行版级别。

**配置类型**

数据库

**参数类型**

参考

在迁移未完成或失败的情况下，此参数将反映未迁移数据库的发行版级别，而且它可能与 *release* 参数（数据库配置文件的发行版级别）不同。否则，*database_level* 的值将等于 *release* 参数的值。

## database_memory - 数据库共享内存大小

此参数指定为数据库共享内存区域保留的内存量。如果此数值小于根据各个内存参数（例如，锁定列表、实用程序堆和缓冲池等）计算而得的数值，那么将使用更大的数值。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

Automatic [Computed, 0 - 4 294 967 295 ]

**计量单位**

页 (4 KB)

**分配时间**
当数据库激活时

**释放时间**
当取消激活数据库时

将此参数设置为 AUTOMATIC 将启用自调整功能。当内存调整器处于启用状态时，它确定数据库的整体内存需求并根据当前数据库需求增大或减小分配给数据库共享内存的内存量。例如，如果当前数据库需求很高，并且系统上有足够的可用内存，那么数据库共享内存将消耗较多的内存。当数据库内存需求下降时，或者当系统上的可用内存量降至过低水平时，就会释放一些数据库共享内存。

内存调整器始终根据计算得到的为实例提供附加内存的好处来保留最低可用内存量。如果为实例提供更多内存的好处很大，内存调整器就会维持较低的可用内存量。如果好处不大，就会维持较高的可用内存量。这样，数据库就可以参与系统内存分配工作。

由于内存调整器在不同内存使用者之间交换内存资源，所以，必须至少有两个内存使用者启用自调整功能才能使自调整功能有效。

仅当启用了数据库的自调整内存功能（**self_tuning_mem** 配置参数设置为 ON）时，才会自动调整此配置参数。

为了简化对此参数的管理，COMPUTED 设置指示数据库管理器计算所需的内存量，并在激活数据库时分配此内存量。数据库管理器还将分配一些附加内存，以便数据库共享内存区域中任何堆超出其配置大小时满足其峰值内存需求。其他操作（例如动态配置更新）也能访问这些附加内存。可以使用带有 **-memsets** 选项的 db2pd 命令来监视数据库共享内存区域中剩余的未使用内存量。

**建议**：此值通常保持为 AUTOMATIC。对于不支持 AUTOMATIC 设置的环境，此参数应设置为 COMPUTED。例如，附加内存可用于创建新的缓冲池或增大现有缓冲池的大小。

**注**:

- 在版本 9.5 中，将 **database_memory** 配置参数设置为 AUTOMATIC 时，分配的初始数据库共享内存是为数据库定义的所有堆和缓冲池的配置大小，并且内存将在需要时增大。如果 **database_memory** 设置为特定值，那么在数据库激活期间最初会分配请求的内存量。如果不能从操作系统分配初始内存量，或初始内存量超过 **instance_memory** 限制，那么数据库激活将失败，并且出现 SQL1084C 错误（不能分配共享内存段）。
- 截至 DB2 版本 9.5 修订包 2，如果在 Solaris Operating Environment 上将 **database_memory** 设置为 AUTOMATIC，那么数据库管理器会将可分页内存用作数据库共享内存。在 UltraSPARC 上的 Solaris 操作系统中，数据库管理器会尝试使用 64 KB 内存页（如果可用）。如果 64 KB 内存页不可用，那么数据库管理器将使用 8 KB 内存页。在 Sun x64 系统上的 Solaris 操作系统中，数据库管理器将使用 4 KB 内存页。使用较小的内存页可能会导致性能有所下降。由于使用了可分页共享内存，因此还需要更多交换空间（等于数据库共享内存大小）。

- 在 Solaris 上，从 DB2 版本 9.5 修订包 2 开始，当 **database_memory** 设置为 COMPUTED 或者数值时，数据库管理器将使用私有的共享内存 (ISM) 和大内存页作为数据库共享内存。在 DB2 版本 9.5 修订包 2 之前，在这种情况下使用的是动态的私有共享内存 (DISM)。

**控制 DB2 内存消耗:**

**instance_memory** 设置为 AUTOMATIC 时，将在实例启动（db2start）时为该实例设置一个固定的总内存消耗上限。数据库管理器实际消耗的内存随工作负载不同而有所变化。启用自调整堆管理器以执行 **database_memory** 调整时（缺省情况下对新数据库启用此功能），自调整内存管理器将在运行时根据系统上的可用物理内存来动态更新数据库共享内存集中性能关键堆的大小，并同时确保有足够的可用 **instance_memory** 用于满足功能内存需求。有关更多信息，请参阅 **instance_memory** 配置参数。

**某些 Linux[1] 内核存在的局限性:**

由于某些 Linux 内核上的操作系统局限性，自调整内存管理器当前不允许将 **database_memory** 设置为 AUTOMATIC。但是目前，只有在 **instance_memory** 设置为特定值而不是 AUTOMATIC 时，才允许在这些内核上使用此设置。如果 **database_memory** 设置为 AUTOMATIC，并且稍后将 **instance_memory** 重新设置为 AUTOMATIC，那么在下一次激活数据库期间 **database_memory** 配置参数将自动更新为 COMPUTED。如果某些数据库已处于活动状态，那么自调整内存管理器将停止调整 **database_memory** 的总大小。

[1]在 Linux 上，此参数支持在 RHEL5 和 SUSE 10 SP1 以及更高版本上使用 AUTOMATIC 设置。如果内核不支持此功能，那么所有其他经验证的 Linux 分发产品将重新使用 COMPUTED。

## db_mem_thresh - 数据库内存阈值

此参数表示数据库管理器允许的已落实但当前未使用的数据库共享内存最大百分比，达到此百分比后，数据库管理器将开始释放已落实的内存页以将它们返回给操作系统。

**配置类型**

数据库

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

10 [0–100 ]

**计量单位**

百分比

此数据库配置参数指定数据库管理器如何处理未用数据库共享内存过多这一问题。通常，当进程访问内存页时，内存页处于已落实状态，这表示操作系统已分配了该内存页，并且该内存页占用物理内存空间或磁盘上的页文件空间。根据数据库的工作负载，数据库共享内存需求在一天中的某些时间可能会达到峰值。一旦操作系统有足够的已落实内存来满足那些峰值需求后，该内存就会一直处于已落实状态，即使内存需求从峰值回落亦如此。

可接受的值是整数 0（立即释放任何未使用的数据库共享内存）到 100（永远不释放任何未使用的数据库共享内存）。缺省值是 10（仅当当前未使用的数据库共享内存超过 10% 时，才释放那些内存），它应该适合大多数工作负载。

可以动态地更新此配置参数。更新此参数时应该十分谨慎，这是因为，将值设置得太小会导致计算机内存负担过大（不断地落实并释放内存页），而将值设置得太大会导致数据库管理器无法将任何数据库共享内存返回给操作系统以供其他进程使用。

如果通过 DB2_PINNED_BP 注册表变量确定了数据库共享内存区域、通过 DB2_LARGE_PAGE_MEM 注册表变量为大内存页配置了数据库共享内存区域或者通过 DB2MEMDISCLAIM 注册表变量显式禁止释放内存，那么将忽略此配置参数（这表示未使用的数据库共享内存页将保持处于已落实状态）。

某些版本的 Linux 不支持将共享内存段的子范围释放回给操作系统。在这样的平台上，将忽略此参数。

## dbheap - 数据库堆

此参数确定数据库堆使用的最大内存。

对于版本 9.5，此数据库配置参数的缺省值为 AUTOMATIC，这表示数据库堆可以根据需要增大，直到达到 *database_memory* 限制或达到 *instance_memory* 限制。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
Automatic [32 - 524 288]

**计量单位**
页 (4 KB)

**分配时间**
当数据库激活时

**释放时间**
当取消激活数据库时

每个数据库有一个数据库堆，并且数据库管理器代表所有连接至数据库的应用程序使用数据库堆。它包含表、索引、表空间和缓冲池的控制块信息。它还包括日志缓冲区的空间（*logbufsz*）和实用程序使用的临时内存。因此堆大小将取决于许多变量。控制块信息保存在堆中，直到所有应用程序与数据库断开连接为止。

启动数据库管理器时需要的最小量是在第一次连接时分配的。数据区将根据需要扩展，直到达到所配置的上限，或者在设置为 AUTOMATIC 的情况下，直到耗尽所有 *database_memory* 和/或 *instance_memory* 内存为止。

决定要对 *dbheap* 配置参数指定的值时，可以使用以下公式作为一个粗略的准则:

```
10K/表空间 + 4K/表 + (1K + 4*使用的扩展数据块)/范围集群表（RCT）
```

您配置的 *dbheap* 值仅表示分配的一部分数据库堆。数据库堆是用于满足全局数据库内存需求的主内存区。它将调整大小，以便除了包括 *dbheap* 值外，还包括启动数据库所需的基本分配值。用于报告内存使用情况的工具（如内存跟踪程序、快照监视器和 db2pd）将报告较大的那个数据库堆的统计信息。不会单独跟踪 *dbheap* 配置参数所表示的分配值。因此，这些工具所报告的数据库堆内存使用情况的统计信息超过为 *dbheap* 参数配置的值是很正常的。

可以使用数据库系统监视器并借助 *db_heap_top*（分配的最大数据库堆）元素来跟踪用于数据库堆的最大内存量。

**注:**

- 工作负载管理（WLM）工作类集和工作操作集存储在数据库堆中。但是，它们消耗的内存非常少。
- 可信上下文、工作负载管理和审计策略信息高速缓存在内存中以便快速处理。此内存是从数据库堆中分配的。因此，用户定义的可信上下文、工作负载管理和审计策略对象将对数据库堆施加更多内存需求。在此情况下，建议您将数据库堆配置设置为 AUTOMATIC，以便数据库管理器将自动管理数据库堆大小。

## decflt_rounding - 十进制浮点数舍入配置参数

此参数允许您指定十进制浮点数（DECFLOAT）的舍入方式。舍入方式影响服务器的 LOAD 中的十进制浮点数操作。

**配置类型**

数据库

**参数类型**

可配置

请参阅下面的第 498 页的『更改 decflt_rounding 的后果』。

**缺省值 [范围]**

ROUND_HALF_EVEN [ROUND_CEILING, ROUND_FLOOR, ROUND_HALF_UP, ROUND_DOWN]

DB2 支持五种符合 IEEE 标准的十进制浮点数舍入方式。舍入方式指定在计算结果超过精度时如何舍入结果。所有舍入方式的定义如下所示:

**ROUND_CEILING**

向正无穷大方向舍入。如果删除的所有位都是零，或者符号为负号，那么结果不变。否则，结果系数应增加 1（向上舍入）。

**ROUND_FLOOR**

向负无穷大方向舍入。如果删除的所有位都是零，或者符号为正号，那么结果不变。否则，如果符号为负号，那么结果系数应增加 1。

**ROUND_HALF_UP**

舍入为最接近的整数；如果向上舍入与向下舍入是等距的，那么向上舍入并使结果系数增加 1。如果被删除的位大于或等于它左边位置中值 1 的一半（0.5），那么结果系数应增加 1（向上舍入）。否则，将忽略被删除的位（小于或等于 0.5）。

**ROUND_HALF_EVEN**

舍入为最接近的整数；如果向上舍入与向下舍入是等距的，那么按照使最后一

位为偶数的目标来进行舍入。如果被废弃的位大于它左边位置中值 1 的一半（0.5），那么结果系数应增加 1（向上舍入）。如果它们小于一半，那么不会调整结果系数，即，将忽略被废弃的位。否则，在它们表示刚好一半的情况下，如果结果系数最右边的一位是偶数，那么结果系数将不会改变；如果结果系数最右边的一位是奇数，那么结果系数应增加 1（向上舍入），以使它变成偶数位。根据 IEEE 十进制浮点数规范，此舍入方式是缺省舍入方式，并且它是 DB2 产品中的缺省舍入方式。

**ROUND_DOWN**
朝着 0 的方向舍入（截断）。将忽略被废弃的位。

表 71 显示了在不同舍入方式下，将 12.341、12.345、12.349、12.355 和 -12.345 均舍入为 4 位的结果:

*表 71. 十进制浮点数舍入方式*

| 舍入方式 | 12.341 | 12.345 | 12.349 | 12.355 | -12.345 |
|---|---|---|---|---|---|
| ROUND_DOWN | 12.34 | 12.34 | 12.34 | 12.35 | -12.34 |
| ROUND_HALF_UP | 12.34 | 12.35 | 12.35 | 12.36 | -12.35 |
| ROUND_HALF_EVEN | 12.34 | 12.34 | 12.35 | 12.36 | -12.34 |
| ROUND_FLOOR | 12.34 | 12.34 | 12.34 | 12.35 | -12.35 |
| ROUND_CEILING | 12.35 | 12.35 | 12.35 | 12.36 | -12.34 |

### 更改 decflt_rounding 的后果

- 先前构造的具体化查询表 (MQT) 包含的结果可能与使用新的舍入方式产生的结果不同。为了更正此问题，请刷新潜在受影响的 MQT。
- 触发器的结果可能受新的舍入方式影响。更改舍入方式不会对已写入的数据产生任何影响。
- 如果重新评估，那么允许将数据插入到表中的约束可能会拒绝相同的数据。如果重新评估，那么不允许将数据插入到表中的类似约束可能会接受相同的数据。使用 SET INTEGRITY 语句来检查并更正这种问题。当生成列值的计算取决于 decflt_rounding 时，如果有两个完全相同的行，一个行是在对 decflt_rounding 进行更改前插入的，而另一个行是在更改后插入的，那么这两行的生成列值可能不同（已生成的列值除外）。
- 舍入方式未编译到段中。因此，在更改 *decflt_rounding* 后，不需要重新编译静态 SQL。

**注:** 此配置参数的值不会动态更改，但它仅在所有应用程序与数据库断开连接后才生效。如果数据库被激活，那么必须将其重新激活。

## dft_degree - 缺省度

此参数指定 CURRENT DEGREE 专用寄存器和 DEGREE 绑定选项的缺省值。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 连接

**缺省值 [范围]**
1 [-1(ANY), 1 - 32 767]

缺省值为 1。

值为 1 则意味着没有分区内并行性。值 -1（或 ANY）意味着优化器根据处理器数目和查询类型确定分区内并行度。

编译语句时使用 CURRENT DEGREE 专用寄存器或 DEGREE 绑定选项指定 SQL 语句的分区内并行度。使用 SET RUNTIME DEGREE 命令指定活动应用程序的最大运行时分区内并行度。“最大查询并行度”（*max_querydegree*）配置参数指定对于所有 SQL 查询的最大查询分区内并行度。

实际使用的运行时并行度是下列值中最小的一个:

- *max_querydegree* 配置参数
- 应用程序运行时并行度
- SQL 语句编译并行度

## dft_extent_sz - 表空间的缺省扩展数据块大小

此参数设置表空间的缺省扩展数据块大小。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
32 [2 – 256 ]

**计量单位**
页

在创建表空间时，可任意地指定 `EXTENTSIZE n`，其中 `n` 为扩展数据块大小。如果不在 CREATE TABLESPACE 语句上指定扩展数据块大小，那么数据库管理器使用由此参数给定的值。

**建议**：在多数情况中，应在创建表空间时明确指定扩展数据块大小。在选择此参数的值之前，应了解如何显式选择 CREATE TABLESPACE 语句的扩展数据块大小。

## dft_loadrec_ses - 缺省装入恢复会话数

此参数指定将在表装入的恢复期间使用的缺省会话数。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
1 [1 - 30 000]

**计量单位**
计数器

应将此值设置为要用来检索装入副本的最佳 I/O 会话数。对装入副本进行检索是与复原相似的操作。也可以通过在环境变量 DB2LOADREC 指定的副本位置文件中的条目来覆盖此参数。

用于装入检索的缺省缓冲区数比此参数的值多两个。也可以覆盖副本位置文件中的缓冲区数。

仅当启用了前滚恢复时，此参数才可应用。

## dft_mttb_types - 为优化缺省维护的表类型

此参数指定 CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 专用寄存器的缺省值。此专用寄存器的值确定在优化查询期间将使用哪些类型的延迟刷新的具体化查询表。

**配置类型**
数据库

**参数类型**
可配置

**缺省值 [范围]**
SYSTEM [ ALL, NONE, FEDERATED_TOOL, SYSTEM, USER 或者值的列表]

可以指定用逗号分隔的值的列表；例如，“USER,FEDERATED_TOOL”。不能将 ALL 或 NONE 与其他值一起列示，并且不能多次指定同一个值。要供 db2CfgSet 和 db2CfgGet API 使用，可接受的参数值为：8（ALL）、4（NONE）、16（FEDERATED_TOOL）、1（SYSTEM）和 2（USER）。可以使用按位或同时指定多个值；例如，18 等价于 USER 和 FEDERATED_TOOL。和以前一样，值 4 和 8 不能与其他值一起使用。

## dft_prefetch_sz - 缺省预取大小

此参数设置表空间的缺省预取大小。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
Automatic [0 - 32767]

**计量单位**
页

创建了表空间之后，可以选择将 PREFETCHSIZE 指定为值 AUTOMATIC 或 *n*，其中 *n* 表示数据库管理器在执行预取时将读取的页数。如果调用 CREATE TABLESPACE 语句时不指定预取大小，那么数据库管理器将使用 **dft_prefetch_sz** 参数的当前值。

如果表空间是在预取大小设置为 AUTOMATIC 的情况下创建的，那么 DB2 数据库管理器将自动计算和更新表空间的预取大小。

在下列情况下执行此计算:

- 当数据库启动时
- 第一次使用 AUTOMATIC 预取大小来创建表空间时
- 当通过执行 ALTER TABLESPACE 语句来更改表空间的容器数目时
- 当通过执行 ALTER TABLESPACE 语句来将表空间的预取大小更新为 AUTOMATIC 时

一旦通过调用 ALTER TABLESPACE 语句手动更新了预取大小，就可以打开或关闭预取大小的 AUTOMATIC 状态。

**建议**: 使用系统监视工具，可以确定当系统等待 I/O 时您的 CPU 是否空闲。如果未定义正使用的表空间的预取大小，那么增大此参数的值可能有所帮助。

此参数提供整个数据库的缺省值，它可能并不适合数据库中的所有表空间。例如，值 32 可能适合扩展数据块大小为 32 页的表空间，但不适合于扩展数据块大小为 25 页的表空间。理想情况下，应显式设置每个表空间的预取大小。

为了帮助将用缺省扩展数据块大小（**dft_extent_sz**）定义的表空间的 I/O 降至最低，应该将此参数设置为 **dft_extent_sz** 参数值的一个因子或整数倍。例如，如果 **dft_extent_sz** 参数是 32，可将 **dft_prefetch_sz** 设置为 16（32 的因子）或 64（32 的整数倍）。如果预取大小是该扩展数据块大小的倍数，那么在下列情况下数据库管理器可以并行执行 I/O:

- 正在预取的扩展数据块在不同的物理设备上
- 配置了多个 I/O 服务器（**num_ioservers**）。

## dft_queryopt - 缺省查询优化类

在编译 SQL 和 XQuery 查询时，查询优化类用于指导优化器使用不同程度的优化。此参数提供了额外的灵活性，这是通过设置在既没有使用 SET CURRENT QUERY OPTIMIZATION 语句也没有在绑定命令上的 QUERYOPT 选项时使用缺省查询优化类实现的。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 连接

**缺省值 [范围]**
5 [ 0 — 9 ]

**计量单位**
查询优化类（请参阅以下内容）

当前定义的查询优化类是:

- 0 - 最小查询优化。
- 1 - 大致与 DB2 V1 相当。
- 2 - 轻微优化。
- 3 - 适度查询优化。
- 5 - 有效的试探查询优化，可限制选择访问方案所消耗的成本。这是缺省情况。
- 7 - 有效的查询优化。
- 9 - 最大查询优化

## dft_refresh_age - 缺省刷新持续时间

此参数表示在特定 REFRESH DEFERRED 具体化查询表上处理 REFRESH TABLE 语句所用的最长持续时间。在超过此时间限制后，具体化查询表将不用来满足查询，直到刷新了具体化查询表为止。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

0 [ 0, 99999999999999 (ANY)]

此参数带有未指定 CURRENT REFRESH AGE 专用寄存器时用于 REFRESH AGE 的缺省值。此参数指定一个时间戳记持续时间值，其数据类型为 DECIMAL(20,6)。如果 CURRENT REFRESH AGE 的值为 99999999999999（ANY），且 QUERY OPTIMIZATION 级别为二、五或更多，那么考虑使用 REFRESH DEFERRED 具体化查询表来优化处理动态查询。

## dft_sqlmathwarn - 出现算术异常时继续

此参数设置缺省值，在 SQL 语句编译期间出现错误或警告时，该缺省值确定处理算术错误和检索转换错误的方法。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

No [No, Yes]

对于静态 SQL 语句，此参数值与绑定时的程序包相关。对于动态 SQL DML 语句，在准备语句时使用此参数的值。

**注意**: 如果您更改数据库的 *dft_sqlmathwarn* 值，那么包括算术表达式的检查约束、触发器和视图的行为也可能会更改。反过来，这也可能会影响数据库的数据完整性。您只应在仔细评估新的算术异常处理行为可以如何影响检查约束、触发器和视图之后，才能为一个数据库更改 *dft_sqlmathwarn* 的设置。一旦更改，接下来的更改同样也需要仔细评估。

例如，考虑下列检查约束，它包括一个除法算术运算:

```
A/B > 0
```

当 *dft_sqlmathwarn* 为“No”且试图执行 B=0 的 INSERT 时，将被零除作为算术错误来处理。由于 DB2 不能检查约束，该插入操作失败。如果将 *dft_sqlmathwarn* 更改为“YES”，那么将被零除作为算术警告来处理，结果为 NULL。结果为 NULL 会导致该谓词求值为 UNKNOWN，从而插入操作能够成功运行。如果将 *dft_sqlmathwarn* 更改回“No”，那么插入同一行的尝试将失败，因为被零除的错误将阻止 DB2 评估该约束。当 *dft_sqlmathwarn* 为“YES”时用 B=0 插入的行将保留在该表中，且可选择。对于导致对约束进行求值的那一行的更新将失败，而对于不需要约束重新求值的那一行的更新则将成功。

在将 *dft_sqlmathwarn* 从“No”更改为“Yes”之前，您应考虑重新编写该约束，以显式处理算术表达式产生的空值。例如:

```
  ( A/B > 0 ) AND ( CASE
                                        WHEN A IS NULL THEN 1
                                        WHEN B IS NULL THEN 1
                                        WHEN A/B IS NULL THEN 0
                                        ELSE 1
                      END                                       = 1 )
```

能在 A 和 B 为可空时使用。如果 A 或 B 为不可空，那么可除去相应的 IS NULL WHEN 子句。

在将 *dft_sqlmathwarn* 从“Yes”更改为“No”之前，您应首先检查是否有可能变得不一致的数据，例如，通过使用如下所示的谓词:

```
WHERE A IS NOT NULL AND B IS NOT NULL AND A/B IS NULL
```

当隔离不一致的行时，在更改 *dft_sqlmathwarn* 之前，您应执行适当的操作以校正不一致性 也可以在更改之后用算术表达式手动复查约束。为此，首先使受到影响的表处于检查暂挂状态（通过 SET CONSTRAINTS 语句的 OFF 子句），然后请求检查这些表（通过 SET CONSTRAINTS 语句的 IMMEDIATE CHECKED 子句）。算术错误将指示不一致的数据，这可避免对约束进行求值。

**建议**: 除非您明确地需要处理包括算术异常的查询，否则，使用缺省设置 no。然后将该值指定为 yes。如果在其他数据库管理器中处理提供结果的 SQL 语句，而不考虑发生的算术异常，那么可能发生此情况。

## discover_db - 发现数据库

在服务器上接收到发现请求时，此参数用来防止将关于数据库的信息返回至客户机。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
Enable [Disable, Enable]

此参数的缺省值为对此数据库启用发现。

通过将此参数值更改为“Disable”，可以隐藏含有敏感数据的数据库，以防止被发现进程发现。此操作可与数据库上的其他数据库安全性控制一起完成。

## dlchktime - 检查死锁的时间间隔

此参数定义数据库管理器检查所有与数据库连接的应用程序间的死锁的频率。

**配置类型**

数据库

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

10 000（10 秒）[1 000 - 600 000]

**计量单位**

毫秒

当连接至同一数据库的两个或多个应用程序无限制地等待一个资源时发生死锁。因为每个应用程序都挂起对方执行所需要的资源，所以该等待永远得不到解决。

**注:**

1. 在分区数据库环境中，此参数只适用于目录节点。
2. 在分区数据库环境中，死锁直到第二次重复出现之后才被标记。

**建议**：增大此参数以降低检查死锁的频率，因此增加应用程序必须等待消除死锁的时间。

减小此参数会增大检查死锁的频率，从而减少应用程序必须等待死锁解决的时间，但是会增加数据库管理器检查死锁所花的时间。如果死锁时间间隔太小，可能会降低运行时性能，因为数据库管理器频繁执行死锁检测。如果将此参数设置得较低以改善并行性，那么应确保适当地设置 *maxlocks* 和 *locklist*，以避免不必要的锁定升级，这种升级可能导致更多的锁定争用，由此产生更多死锁的情况。

## dyn_query_mgmt - 动态 SQL 和 XQuery 查询管理

此参数确定 Query Patroller 是否将捕获有关已提交的查询的信息。

**配置类型**

数据库

**参数类型**

可联机配置

**缺省值 [范围]**

0 (DISABLE) [ 1(ENABLE), 0 (DISABLE) ]

此参数与 DB2 Query Patroller 的安装位置相关。如果此参数设置为“ENABLE”，那么 Query Patroller 将捕获有关查询的信息，例如，提交者标识和估计的执行成本（由优化器计算）。这些值用来确定是否应该由 Query Patroller 根据用户级别阈值和系统级别阈值来管理查询。

如果此参数设置为“DISABLE”，那么 Query Patroller 不会捕获有关提交的查询的任何信息，也不会进行查询管理。

## enable_xmlchar - 对 XML 启用配置参数转换

此参数确定是否可以对 SQL 语句中的非 BIT DATA CHAR（或 CHAR 类型）表达式执行 XMLPARSE 操作。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

Yes [ Yes; No ]

在非 Unicode 数据库中使用 pureXML® 功能时，XMLPARSE 函数可能会导致发生字符替换，因为 SQL 字符串数据会从客户机代码页转换为数据库代码页，然后转换为 Unicode 以进行内部存储。将 *enable_xmlchar* 设置为 NO 将阻止在 XML 解析期间使用字符数据类型，并且任何将字符类型插入到非 Unicode 数据库中尝试都会生成错误。当 *enable_xmlchar* 设置为 NO 时，仍然允许使用 BLOB 数据类型和 FOR BIT DATA 数据类型，这是因为使用这些数据类型来将 XML 数据传递到数据库中时不会进行代码页转换。

缺省情况下，*enable_xmlchar* 设置为 YES，以便允许解析字符数据类型。在这种情况下，应确保要插入的任何 XML 数据仅包含作为数据库代码页的一部分的代码点，以避免在插入 XML 数据期间引入替换字符。

**注**：客户机需要断开与代理程序的连接，然后重新连接才能反映此更改。

## failarchpath - 故障转移日志归档路径

此参数指定一条路径，如果由于影响归档目标的介质问题，而不能将日志文件归档至主要归档目标或辅助归档目标（如果设置了此项），那么 DB2 将尝试按此路径对日志文件进行归档。指定的此路径必须引用一个磁盘。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

Null [ ]

如果 *failarchpath* 的当前值所指定的路径中存在日志文件，那么对 *failarchpath* 进行的任何更新都不会立即生效。相反，更新将在所有应用程序断开连接后生效。

## groupheap_ratio - 用于应用程序组堆的内存百分比

在版本 9.5 中不推荐使用此参数，但是版本 9.5 之前的数据服务器和客户机仍然使用此参数。DB2 V9.5 数据库管理器将忽略对此配置参数指定的任何值。在 V9.5 中，它已被 *appl_memory* 配置参数取代。

**注:** 下列信息仅适用于版本 9.5 之前的数据服务器和客户机。

此参数指定用于应用程序组共享堆的应用程序控制共享内存集中的内存百分比。

**配置类型**
数据库

**参数类型**
可配置

**缺省值 [范围]**
70 [1 – 99 ]

**计量单位**
百分比

此参数对于集中器关闭且禁用分区内并行性的非分区数据库不起任何作用。

**建议:** 除非遇到性能问题，否则请保留此参数的缺省值。

## hadr_db_role - HADR 数据库角色

此参数指示数据库的当前角色，不管数据库是处于联机还是脱机状态。

**配置类型**
数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器

**参数类型**
参考

有效值为: STANDARD、PRIMARY 或 STANDBY。

**注:** 当数据库处于活动状态时，还可以使用 GET SNAPSHOT FOR DATABASE 命令确定数据库的 HADR 角色。

## hadr_local_host - HADR 本地主机名

此参数指定用于高可用性灾难恢复 (HADR)TCP 通信的本地主机。

**配置类型**
数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器

**参数类型**

可配置

**缺省值** Null

可以使用主机名也可以使用 IP 地址。如果指定了主机名，但是它映射至多个 IP 地址，那么会返回错误并且将不启动 HADR。如果主机名映射至多个 IP 地址（即使对主数据库和备用数据库指定同一个主机名，它也映射至多个 IP 地址），那么主数据库和备用数据库会结束将此主机名映射至不同的 IP 地址，原因是一些 DNS 服务器将按不确定的顺序返回 IP 地址列表。

主机名的格式为: myserver.ibm.com。IP 地址的格式为: “12.34.56.78”。

## hadr_local_svc - HADR 本地服务名称

此参数指定本地高可用性灾难恢复 (HADR) 进程将接受连接的 TCP 服务名称或端口号。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器

**参数类型**

可配置

**缺省值** Null

主数据库系统或备用数据库系统上的 hadr_local_svc 值不能与它们各自节点上的 svcename 或 svcename +1 值相等。

## hadr_peer_window - HADR 对等窗口配置参数

如果将 **hadr_peer_window** 设置为一个非 0 时间值，那么在主数据库与备用数据库断开连接的情况下，HADR 主数据库/备用数据库对在所配置的时间段内仍将表现为处于对等状态一样。这可帮助确保数据的一致性。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

0 [0 – 4 294 967 295]

**计量单位**

秒

**使用说明:**

- 该值在主数据库和备用数据库上需要相同。
- 建议最小值为 120 秒。

- 当 **hadr_syncmode** 的值设置为 ASYNC 时，将忽略 **hadr_peer_window** 的值。也就是说，会将该值当作设置为 0 来对待，因为它在 ASYNC 方式下没有意义。
- 为了避免在故意关闭备用数据库（例如，为了进行维护）时影响主数据库的可用性，那么如果在 HADR 对处于对等状态时显式取消激活了备用数据库，就不会调用对等窗口。
- 仅当 HADR 备用当前在定义的对等窗口内时，执行带有 **PEER WINDOW ONLY** 选项的 TAKEOVER HADR 命令才会启动接管操作。
- 如果主数据库时钟与备用数据库时钟在 5 秒钟之内未实现彼此同步，那么使用 **PEER WINDOW ONLY** 选项的接管操作的行为可能会不正确。也就是说，当接管操作应失败时它却成功了，而应该成功时却失败了。应使用时间同步服务（例如，NTP）来使上述两个时钟都与同一个源同步。
- 在备用数据库中，对等窗口结束时间取决于上次从主数据库接收到脉动信号消息的时间而不是断开连接的时间。因此，在转换为 S-RemoteCatchupPending 状态之前，备用数据库处于 S-DisconnectedPeer 状态的剩余时间范围为从（**hadr_peer_window** - **hadr_timeout**）到（**hadr_peer_window**）秒，具体取决于断开连接所发生的时间和方式。

## hadr_remote_host - HADR 远程主机名

此参数指定远程高可用性灾难恢复 (HADR) 数据库服务器的 TCP/IP 主机名或 IP 地址。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器

**参数类型**

可配置

**缺省值** Null

与 *hadr_local_host* 相似，此参数必须只映射至一个 IP 地址。

## hadr_remote_inst - 远程服务器的 HADR 实例名

此参数指定远程服务器的实例名称。一些管理工具（例如，DB2 控制中心）使用此参数来与远程服务器联系。高可用性灾难恢复 (HADR) 还检查请求连接的远程数据库是否属于声明的远程实例。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器

**参数类型**

可配置

缺省值 Null

## hadr_remote_svc - HADR 远程服务名称

此参数指定远程高可用性灾难恢复 (HADR) 数据库服务器将使用的 TCP 服务名称或端口号。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器

**参数类型**

可配置

**缺省值** Null

## hadr_syncmode - 处于对等状态的日志写操作的 HADR 同步方式

此参数指定同步方式，它确定当系统处于对等状态时主日志写操作如何与备用日志写操作同步。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

NEARSYNC [ASYNC; SYNC ]

此参数的有效值为:

**SYNC** 此方式对防止发生事务丢失提供了最强有力的保护，但是事务响应时间较长。

在此方式中，仅当日志已写入主数据库上的日志文件，而且主数据库已接收到来自备用数据库的应答，确定日志也已写入备用数据库上的日志文件时，方才认为日志写入是成功的。保证日志数据同时存储在这两处。

**NEARSYNC**

此方式对防止发生事务丢失提供的保护稍微差一点，但是，与 SYNC 方式比较起来，NEARSYNC 方式的事务响应时间较短。

在此方式中，仅当日志记录已写入主数据库上的日志文件，而且主数据库已接收到来自备用系统的应答，确定日志也已写入备用系统上的主存储器时，方才认为日志写入是成功的。仅当两处同时发生故障，并且目标位置未将接收到的所有日志数据转移至非易失性存储器时，才会出现数据的丢失。

**ASYNC**

在主数据库发生故障的情况下，此方式发生丢失事务的可能性最大，但是，在这三种方式中，该方式的事务响应时间最短。

在此方式中，仅当日志记录已写入主数据库上的日志文件，而且已将此记录传递给主系统主机的 TCP 层时，方才认为日志写入是成功的。因为主系统不会等待来自备用系统的应答，所以当事务仍处于正在传入备用系统的过程中时，可能会认为事务已落实。

## hadr_timeout - HADR 超时值

此参数指定在认为尝试通信失败之前高可用性灾难恢复 (HADR) 进程要等待的时间（以秒计）。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

120 [1 - 4 294 967 295 ]

## indexrec - 索引重新创建时间

此参数指示数据库管理器将尝试重建无效索引的时间，以及在 DB2 前滚期间或在备用数据库上重放 HADR 日志期间是否重做任何索引构建。

**配置类型**

数据库和数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

**UNIX 数据库管理器**

restart [ restart; restart_no_redo; access; access_no_redo ]

**Windows 数据库管理器**

restart [ restart; restart_no_redo; access; access_no_redo ]

**数据库** 使用系统设置 [system; restart; restart_no_redo; access; access_no_redo ]

此参数有五个可能的设置:

**SYSTEM**

在数据库管理器配置文件中指定的 *use system setting* 决定何时重建无效的索引,

以及在 DB2 前滚或 HADR 日志重放期间是否将重做任何索引构建日志记录。（注意：此设置仅对数据库配置有效。）

**ACCESS**

第一次访问索引时将重建无效的索引。在 DB2 前滚或 HADR 日志重放期间将重做任何完全记录的索引构建。当启动了 HADR 且发生 HADR 接管时，将在接管之后第一次访问基础表时重建任何无效的索引。

**ACCESS_NO_REDO**

第一次访问基础表时将重建无效的索引。在 DB2 前滚或 HADR 日志重放期间将不重做任何完全记录的索引构建，那些索引将保留为无效状态。当启动了 HADR 且发生 HADR 接管时，将在接管之后第一次访问基础表时重建任何无效的索引。

**RESTART**

*indexrec* 的缺省值。将在显式或隐式发出 RESTART DATABASE 命令时重建无效的索引。在 DB2 前滚或 HADR 日志重放期间将重做任何完全记录的索引构建。当启动了 HADR 且发生 HADR 接管时，将在接管结束时重建任何无效的索引。

注意，如果启用 *autorestart* 参数，那么隐式发出 RESTART DATABASE 命令。

**RESTART_NO_REDO**

将在显式或隐式发出 RESTART DATABASE 命令时重建无效的索引。（如果启用了 *autorestart* 参数，将隐式地发出 RESTART DATABASE 命令。）在 DB2 前滚或 HADR 日志重放期间将不重做任何完全记录的索引构建，而是在前滚完成时或在 HDAR 接管发生时将重建那些索引。

当出现严重的磁盘问题时，索引可能会变为无效。如果数据本身出现这个问题，那么数据可能丢失。但是，如果索引出现这个问题，那么可通过重新创建该索引来恢复索引。如果在用户连接至数据库时重建索引，那么可能出现两个问题:

- 重新创建索引文件时可能会发生响应时间意外降低现象。访问表和使用此特定索引的用户将在重建索引时等待。
- 在重新创建索引之后可能挂起意外的锁定，尤其是导致索引重新创建的用户事务从未执行过 COMMIT 或 ROLLBACK 的情况下更是如此。

**建议**：在高端用户服务器上且如果重新启动时间不重要，那么此选项的最佳选择将是在 DATABASE RESTART 时重建该索引，以作为在崩溃后重新将该数据库联机的过程的一部分。

将此参数设置为“ACCESS”或“ACCESS_NO_REDO”将导致重新创建索引时数据库管理器的性能降级。任何访问该特定索引或表的用户将必须等待，直到重新创建索引为止。

如果将此参数设置为“RESTART”，那么重新启动数据库所花的时间将因重新创建索引而较长，但是一旦数据库恢复联机，正常处理将不受影响。

**注**：在进行数据库恢复时，将除去可在属于正在恢复的数据库的文件系统上执行的所有 SQL 过程。如果 *indexrec* 设置为 RESTART，那么会从数据库目录中抽取所有 SQL 过程可执行文件，并在下一次连接至数据库时放回到该文件系统上。如果未将 *indexrec* 设置为 RESTART，那么仅在第一次执行该 SQL 过程时才会将 SQL 可执行文件抽取到该文件系统上。

仅当对索引构建操作（如 CREATE INDEX 和 REORG INDEX 操作）或对索引重建操作激活了完全日志记录时，RESTART 与 RESTART_NO_REDO 值或 ACCESS 与 ACCESS_NO_REDO 值之间的差异才会很明显。可以通过启用 *logindexbuild* 数据库配置参数或者通过在改变表时启用 LOG INDEX BUILD 来激活日志记录。通过将 *indexrec* 设置为 RESTART 或 ACCESS，就可以前滚涉及记录的索引构建的操作而不会使索引对象处于无效状态，如果索引对象处于无效状态，那么需要以后重建该索引。

## jdk_64_path - 64 位 Java 软件开发者工具箱安装路径 DAS

此参数指定要用于运行 DB2 管理服务器函数的 64 位 Java 软件开发者工具箱（SDK）的安装目录。

**配置类型**
DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
Null [任何有效路径]

**注:** 此参数与 **jdk_path** 配置参数不同，后者指定 32 位的 Java SDK。

Java 解释器使用的是根据此参数的环境变量计算出来的。此参数仅在那些既支持 32 位实例又支持 64 位实例的平台上使用。

这些平台包括:

- AIX、HP-UX 和 Solaris 操作系统的 64 位内核
- 64 位的 Windows on X64 和 IPF
- 64 位 Linux kernel on AMD64 以及 Intel® EM64T 系统（x64）、POWER(TM) 和 zSeries®。

在所有其他平台上，只使用 **jdk_path**。

因为此参数没有缺省值，所以安装 Java SDK 时应指定值。

此参数只能从版本 8 命令行处理器（CLP）更新。

## locklist -“锁定列表的最大存储量”

此参数指示分配给锁定列表的内存量。每个数据库有一个锁定列表，锁定列表包含由同时连接至数据库的所有应用程序挂起的锁定。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**

**UNIX** Automatic [4 - 524 288]

**带有本地和远程客户机的 Windows 数据库服务器**
Automatic [4 - 524 288]

**带有本地客户机的 Windows 64 位数据库服务器**
Automatic [4 - 524 288]

**带有本地客户机的 Windows 32 位数据库服务器**
Automatic [4 - 524 288]

**计量单位**
页 (4 KB)

**分配时间**
当第一个应用程序连接至数据库时

**释放时间**
当最后一个应用程序与数据库断开连接时

锁定是数据库管理器用来控制多个应用程序并发访问数据库中的数据的机制。行和表都可以锁定。数据库管理器还可以获取锁定来供内部使用。

当此参数设置为 AUTOMATIC 时，就启用了自调整功能。这允许内存调整器根据工作负载要求变化动态地调整此参数控制的内存区大小。由于内存调整器在不同内存使用者之间交换内存资源，所以，必须至少有两个内存使用者启用自调整功能才能使自调整功能有效。

*locklist* 值是与 *maxlocks* 参数一起调整的，因此，如果禁用 *locklist* 参数自调整功能，也将自动禁用 *maxlocks* 参数自调整功能。如果启用 *locklist* 参数自调整功能，也将自动启用 *maxlocks* 参数自调整功能。

仅当启用了数据库的自调整内存功能（*self_tuning_mem* 配置参数设置为“ON”）时，才会自动调整此配置参数。

在 32 位平台上，每个锁需要 48 或 96 字节的锁定列表，这取决于是否对该对象挂起了其他锁:

- 对于没有挂起其他锁的对象，挂起一个锁需要 96 字节
- 对于已经挂起了锁的对象，记录一个锁需要 48 字节。

在 64 位平台（HP-UX 除外）上，每个锁需要 64 或 128 字节的锁定列表，这取决于是否对该对象挂起了其他锁:

- 对于未挂起其他锁的对象，挂起一个锁需要 128 字节
- 对于存在一个挂起的锁的对象，记录一个锁需要 64 字节。

在 64 位 HP-UX 上，每个锁需要 80 或 160 字节的锁定列表，这取决于是否对该对象挂起了其他锁。

当一个应用程序使用的锁定列表百分比达到 *maxlocks* 时，数据库管理器就会对该应用程序挂起的锁定执行从行到表的锁定升级。虽然升级过程本身花不了多少时间，但是锁定整个表（与个别行比较）降低了并行性，并且可能因对受影响的表进行后续访问而降低整个数据库性能。关于如何控制锁定列表大小的建议是:

- 经常执行 COMMIT 以释放锁定。

- 当执行很多更新时，在更新前锁定整个表（使用 SQL LOCK TABLE 语句）。这样将只使用一个锁定，防止其他锁定干扰更新，但却降低了数据的并行性。

  还可以使用 ALTER TABLE 语句的 LOCKSIZE 选项来控制如何对特定表进行锁定。

  使用“可重复读”隔离级别可能会导致自动表锁定。
- 只要有可能，使用“游标稳定性”隔离级别以减少所挂起的共享锁定数。如果不会影响到应用程序的完整性需求，那么使用“未落实的读”代替“游标稳定性”以进一步减少锁定量。
- 将 *locklist* 设置为 AUTOMATIC。锁定列表将同步地增大以避免发生锁定升级或锁定列表满的情况。

一旦锁定列表已满，性能就可能会降低，因为锁定升级将生成更多的表锁定和更少的行锁定，从而降低数据库中共享对象的并行性。另外，应用程序间可能有更多的死锁（因为这些应用程序都在等待有限数目的表锁定），这样将导致事务回滚。当数据库的锁定请求数达到最大值时，应用程序将接收到值为 -912 的 SQLCODE。

**建议**：如果锁定升级导致性能问题，那么可能需要增大此参数或 *maxlocks* 参数的值。可以使用数据库系统监视器来确定是否正在发生锁定升级。参阅 *lock_escals（锁定升级）*监视元素。

下列步骤可以帮助确定锁定列表所需的页数:

1. 根据您的环境，使用下列计算的*其中一种*计算来为锁定列表的大小计算下限:

   a.
   ```
   (512 * x * maxappls) / 4096
   ```
   b. 启用集中器:
   ```
   (512 * x * max_coordagents) / 4096
   ```
   c. 在启用集中器的分区数据库中:
   ```
   (512 * x * max_coordagents * number of database partitions) / 4096
   ```

   其中 512 是每个应用程序的平均锁定数目的估计值，而 x 是针对具有现有锁定的对象的每个锁定所需要的字节数（在 32 位平台上需要 40 位，那么 64 位平台上需要 64 位）。
2. 计算锁定列表大小的上限:
   ```
   (512 * y * maxappls) / 4096
   ```

   其中 y 是针对对象的第一个锁定所需的字节数（在 32 位平台上需要 80 位，在 64 位平台上需要 128 位）。
3. 估计数据将发生的并行性数量，并根据您的期望为 *locklist* 选择一个在您计算的上限和下限之间的初始值。
4. 按如下所述，使用数据库系统监视器调整此参数的值。

如果在某个可行方案中将 *maxappls* 或 *max_coordagents* 设置为 AUTOMATIC，那么也应该将 *locklist* 设置为 AUTOMATIC。

可以使用数据库系统监视器来确定给定的事务挂起的最大锁定数。参阅 *locks_held_top（挂起的最大锁定数）*监视元素。

此信息可帮助您验证或调整每个应用程序的估计锁定数。为了执行此验证，将必须对几个应用程序进行采样，记下在事务级别而不是应用程序级别上提供的监视信息。

如果增大了 *maxappls*，或者如果正运行的应用程序执行的落实不频繁，那么可能也要增大 *locklist*。

在更改此参数后，应考虑重新绑定应用程序（使用 REBIND 命令）。

## locktimeout - 锁定超时

此参数指定应用程序为获取一个锁定将等待的秒数，以帮助避免应用程序出现全局死锁。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

-1 [-1; 0 - 32 767 ]

**计量单位**

秒

如果将此参数设置为 0，那么不会等待锁定。在此情况下，如果请求时未提供任何锁定，那么应用程序就会立即接收到 -911。

如果将此参数设置为 -1，那么锁定超时检测关闭。在此情况下，将等待锁定（如果请求时无可用锁定），直到出现下列其中一种情况:

- 授予锁定
- 发生死锁。

**建议**: 在事务处理（OLTP）环境中，可以使用 30 秒的初始启动值。在一个只查询的环境中，您可以从一个较高的值开始。在这两种情况下，您都应该使用基准程序技术来调整此参数。

此值应该设置为能够快速检测由于异常情况如事务停止（可能是因为用户离开工作站）而发生的等待。您应该将此值设置得足够大，以便有效锁定请求不会因为峰值工作负载（在该时期内需要更多的锁定等待）而超时。

可以使用数据库系统监视器来帮助您跟踪一个应用程序（连接）经历的锁定超时的次数，或跟踪数据库检测到连接的所有应用程序超时情况的次数。

*lock_timeout*（锁定超时的数目）监视元素的高值可能是下列原因造成的:

- 此配置参数的值太低。
- 挂起锁定较长时间的应用程序（事务）。可以使用数据库系统监视器来进一步调查这些应用程序。
- 并行性问题，它可能是由锁定升级（从行级别至表级别锁定）引起的。

## log_retain_status - 日志保留状态指示器

如果设置了此参数（当 *logretain* 参数设置为 Recovery 时），那么它指示日志文件被保留以用于前滚恢复。

**配置类型**

数据库

**参数类型**

参考

## logarchmeth1 - 主日志归档方法

此参数指定已归档日志的主要目标的介质类型。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

OFF [LOGRETAIN, USEREXIT, DISK, TSM, VENDOR]

**OFF** 指定不使用日志归档方法。如果 **logarchmeth1** 和 **logarchmeth2** 都设置为 OFF，那么认为数据库正在使用循环日志记录，且不可前滚恢复。这是缺省情况。

**LOGRETAIN**

此值仅可用于 **logarchmeth1**，且等价于将 **logretain** 配置参数设置为 RECOVERY。如果指定此值，将自动更新 **logretain** 配置参数。

**USEREXIT**

此值仅对 **logarchmeth1** 有效，且等价于将 **userexit** 配置参数设置为 ON。如果指定此值，将自动更新 **userexit** 配置参数。

**DISK** 此值后必须紧跟冒号（:），然后是现有标准路径名，日志文件将在其中归档。例如，如果将 **logarchmeth1** 设置为 DISK:/u/dbuser/archived_logs，那么将归档日志文件放入名为 /u/dbuser/archived_logs 的目录。

**注**: 如果正在归档至磁带，可以使用 db2tapemgr 实用程序来存储和检索日志文件。

**TSM** 如果指定不带任何附加配置参数，此值指示应该使用缺省管理类，将日志文件归档在本地 TSM 服务器上。如果此值后紧跟冒号（:）和 TSM 管理类，那么使用指定的管理类来归档日志文件。

当通过 TSM 归档日志时，在使用由数据库配置参数指定的管理类之前，TSM 首先尝试将对象绑定到在 TSM 客户机选项文件里的包含/排

除列表中指定的管理类。如果找不到匹配项，那么将使用 TSM 服务器上指定的缺省 TSM 管理类。然后，TSM 会将对象重新绑定到数据库配置参数指定的管理类。

因此，缺省管理类以及数据库配置参数指定的管理类必须包含归档副本组，否则归档操作将失败。

**VENDOR**

指定将使用供应商库来归档日志文件。此值后必须紧跟冒号（:）和库的名称。库中提供的 API 必须使用备份并复原供应商产品的 API。

**注意:**

1. 如果将 **logarchmeth1** 或 **logarchmeth2** 设置为 OFF 以外的值，那么必须配置数据库以进行前滚恢复。
2. 启用或禁用归档日志记录是一项脱机数据库活动。必须在终止与数据库的所有连接之后，新设置才会生效。只能联机执行日志记录路径中的更改。
3. 如果更新 **userexit** 或 **logretain** 配置参数，将自动更新 **logarchmeth1**，反之亦然。然而，如果您要使用 **userexit** 或 **logretain**，必须将 **logarchmeth2** 设置为 OFF。

## logarchmeth2 - 辅助日志归档方法

此参数指定已归档日志的辅助目标的介质类型。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

OFF [ LOGRETAIN, USEREXIT, DISK, TSM, VENDOR]

**OFF** 指定不使用日志归档方法。如果 *logarchmeth1* 和 *logarchmeth2* 都设置为 OFF，那么认为数据库正在使用循环日志记录，且不可前滚恢复。这是缺省情况。

**LOGRETAIN**

此值仅可用于 *logarchmeth1*，且等价于将 *logretain* 配置参数设置为 RECOVERY。如果指定此值，将自动更新 *logretain* 配置参数。

**USEREXIT**

此值仅对 *logarchmeth1* 有效，且等价于将 *userexit* 配置参数设置为 ON。如果指定此值，将自动更新 *userexit* 配置参数。

**DISK** 此值后必须紧跟冒号（:），然后是现有标准路径名，日志文件将在其

中归档。例如，如果将 *logarchmeth1* 设置为 DISK:/u/dbuser/archived_logs，那么将归档日志文件放入名为 /u/dbuser/archived_logs 的目录。

**注**: 如果正在归档至磁带，可以使用 db2tapemgr 实用程序来存储和检索日志文件。

**TSM** 如果指定不带任何附加配置参数，此值指示应该使用缺省管理类，将日志文件归档在本地 TSM 服务器上。如果此值后紧跟冒号（:）和 TSM 管理类，那么使用指定的管理类来归档日志文件。

**VENDOR**
指定将使用供应商库来归档日志文件。此值后必须紧跟冒号（:）和库的名称。库中提供的 API 必须使用备份并复原供应商产品的 API。

**注意:**

1. 如果将 *logarchmeth1* 或 *logarchmeth2* 设置为 OFF 以外的值，那么必须配置数据库以进行前滚恢复。
2. 如果更新 *userexit* 或 *logretain* 配置参数，将自动更新 *logarchmeth1*，反之亦然。但是，如果您要使用 *userexit* 或 *logretain*，必须将 *logarchmeth2* 设置为 OFF。

如果指定了此路径，那么日志文件将被同时归档至此目标和由 *logarchmeth1* 数据库配置参数指定的目标。

## logarchopt1 - 主日志归档选项

此参数为已归档日志的主要目标指定选项字段（如果需要）。

**配置类型**
数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**
可联机配置

**缺省值 [范围]**
Null [不适用]

**限制** 在所配置的支持代理节点的 TSM 环境中，“-fromnode=nodename”选项和“-fromowner=ownername”选项都与“-asnodename=nodename”选项不兼容，不能将它们一起使用。将 -asnodename 选项用于使用代理节点的 TSM 配置，而将另外两个选项用于其他类型的 TSM 配置。有关更多信息，请参阅『配置 Tivoli Storage Manager 客户机』。

## logarchopt2 - 辅助日志归档选项

此参数为已归档日志的辅助目标指定选项字段。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

Null [不适用]

**限制** 在所配置的支持代理节点的 TSM 环境中，“-fromnode=nodename”选项和“-fromowner=ownername”选项都与“-asnodename=nodename”选项不兼容，不能将它们一起使用。将 -asnodename 选项用于使用代理节点的 TSM 配置，而将另外两个选项用于其他类型的 TSM 配置。有关更多信息，请参阅『配置 Tivoli Storage Manager 客户机』。

## logbufsz - 日志缓冲区大小

在将日志记录写入磁盘之前，此参数允许您指定用作这些记录的缓冲区的数据库堆大小（由 *dbheap* 参数定义）。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

**32 位平台**

8 [4 - 4 096 ]

**64 位平台**

8 [4 - 65 535 ]

从版本 9.5 修订包 3 开始，此范围已增加到 8 [4 - 131 070 ]

**计量单位**

页 (4 KB)

当发生下列一种情况时会将日志记录写入磁盘:

- 一个事务落实或一组事务落实，由 *mincommit* 配置参数定义
- 日志缓冲区已满
- 发生了某些其他内部数据库管理器事件。

此参数也必须小于或等于 *dbheap* 参数。缓冲日志记录将使日志记录文件 I/O 更有效，因为将日志记录写入磁盘的频率越低，那么每次可写入的日志记录就越多。

**建议**: 如果在专用的日志磁盘上有大量的读取活动，或者频繁使用磁盘，那么要增大此缓冲区的大小。当增大此参数的值时，您也应考虑 *dbheap* 参数，因为该日志缓冲区使用由 *dbheap* 参数控制的空间。

可以使用数据库系统监视器来确定将多少日志缓冲区空间用于特定事务（或工作单元）。参阅 *log_space_used*（使用的工作单元日志空间）监视元素。

## logfilsiz - 日志文件大小

此参数定义每个主日志文件和辅助日志文件的大小。在这些日志文件已满且需要新日志文件之前，这些日志文件的大小限制可写入这些日志文件的日志记录数。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

**UNIX** 1000 [4 - 524 286]

从版本 9.5 修订包 3 开始，此范围已增加到 1000 [4 - 1 048 572]

**Windows**

1000 [4 - 524 286]

从版本 9.5 修订包 3 开始，此范围已增加到 1000 [4 - 1 048 572]

**计量单位**

页 (4 KB)

主日志文件和辅助日志文件的使用以及在日志文件已满时进行的操作取决于正在执行的日志记录的类型:

- 循环日志记录

  当记录在主日志文件中的更改已落实后，可复用该主日志文件。如果日志文件大小较小，并且应用程序已处理了大量对数据库的更改但未落实这些更改，主日志文件可能会很快变满。如果所有主日志文件变满，那么数据库管理器将分配辅助日志文件来保存新的日志记录。
- 日志保留日志记录

  当主日志文件已满时，将该日志归档并分配新的主日志文件。

**建议**: 必须使日志文件的大小与主日志文件数平衡:

- 如果数据库要运行大量更新、删除或插入事务，而这将导致日志文件很快变满，那么应增大 *logfilsiz* 的值。

  **注**: 日志文件大小的上限与日志文件数目的上限一起 (*logprimary* + *logsecond*) 给出活动日志空间的上限 512 GB。从版本 9.5 修订包 3 开始，可用活动日志空间为 1024 GB。

  日志文件太小则会因归档旧日志文件、分配新日志文件以及等待可用的日志文件的开销而影响系统性能。
- 如果磁盘空间不足，那么应减小 *logfilsiz* 的值，因为主日志是按此大小预分配的。

太大的日志文件会减小管理归档日志文件和日志文件副本时的灵活性，因为某些媒体可能无法保存整个日志文件。

如果正在使用日志保留，那么当最后一个应用程序与数据库断开连接时，关闭并截断当前的活动日志文件。下次与数据库连接时使用下一个日志文件。因此，如果了解您的并发应用程序的日志记录要求，那么可确定一个将不会分配过量浪费空间的日志文件大小。

## loghead - 第一个活动日志文件

此参数包含当前活动的日志文件的名称。

**配置类型**

数据库

**参数类型**

参考

## logindexbuild - 创建的日志索引页数

此参数指定是否要记录索引创建、重新创建或重组操作，以便可以在 DB2 前滚操作或高可用性灾难恢复 (HADR) 日志重放过程中重构索引。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

Off [On; Off ]

## logpath - 日志文件的位置

此参数包含用于进行日志记录的当前路径。

**配置类型**

数据库

**参数类型**

参考

当对 *newlogpath* 参数的更改生效后，您不能直接更改此参数，因为它是由数据库管理器设置的。

当创建一个数据库时，在包含该数据库的目录的一个子目录中创建该数据库的恢复日志文件。缺省值是为该数据库创建的目录下面的子目录 SQLOGDIR。

## logprimary - 主日志文件数

此参数允许您指定要预分配的主日志文件数。主日志文件建立分配给恢复日志文件的固定存储器数量。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

3 [ 2 - 256 ]

**计量单位**

计数器

**分配时间**

- 创建数据库时
- 将日志移至一个不同位置时（当更新 *logpath* 参数时会发生这种情况）
- 下一次启动数据库时，如果此数据库不是作为 HADR 备用数据库启动的，那么此参数 (*logprimary*) 的值将增大
- 已将一个日志文件归档，并分配了一个新日志文件时（必须启用 *logretain* 或 *userexit* 参数）
- 如果已更改 *logfilsiz* 参数，那么在下一次启动数据库时，如果此数据库不是作为 HADR 备用数据库启动的，那么就会调整日志文件的大小。

**释放时间**

除非此参数减小，否则不释放。如果此参数减小，那么在下一个数据库连接期间删除不需要的日志文件。

在循环日志记录下，按顺序重复使用主日志。即，当日志已满时，使用序列中的下一个主日志（如果该主日志可用）。如果已落实或回滚日志中具有日志记录的所有工作单元，那么认为该日志是可用的。如果序列中下一个主日志不可用，那么分配并使用一个辅助日志。分配并使用附加的辅助日志，直到序列中下一个主日志变为可用或达到 *logsecond* 参数所设置的限制为止。当数据库管理器不再需要这些辅助日志文件时，动态释放这些辅助日志文件。

主日志文件和辅助日志文件的数目必须与下列内容一致:

- 如果 *logsecond* 的值为 -1，那么 *logprimary* 小于等于 256。
- 如果 *logsecond* 的值不是 -1，那么 (*logprimary* + *logsecond*) 小于等于 256。

**建议**: 为此参数选择的值取决于许多因素，包括正在使用的日志记录类型、日志文件的大小和处理环境的类型（例如，事务的长度和落实的频率）。

增大此值将增大日志的磁盘需求，因为主日志文件就是在第一个与数据库的连接期间预分配的。

如果您发现经常分配辅助日志文件，那么可通过增大日志文件大小 (*logfilsiz*) 或增大主日志文件的数目来提高系统性能。

对于不经常访问的数据库，为了节省磁盘存储空间，将该参数设置为 2。对于为前滚恢复而启用的数据库，将此参数设置得大一些，以避免几乎立即就分配新日志的开销。

可以使用数据库系统监视器来帮助您确定主日志文件的大小。在一段时间内对下列监视器值的观察将有助于更好的调整决定，因为平均值更能代表当前需求。

- *sec_log_used_top*（使用的最大辅助日志空间）
- *tot_log_used_top*（使用的总日志空间）
- *sec_logs_allocated*（当前分配的辅助日志）

## logretain -"启用日志保留"

在版本 9.5 中不推荐使用此参数，但是版本 9.5 之前的数据服务器和客户机仍然使用此参数。DB2 V9.5 数据库管理器将忽略对此配置参数指定的任何值。从 V9.5 开始，**logarchmeth1** 配置参数用于设置 LOGRETAIN 值。

**注:** 下列信息仅适用于版本 9.5 之前的数据服务器和客户机。

此参数确定是否保留活动日志文件以及这些文件是否可用于前滚恢复。

**配置类型**
数据库

**参数类型**
可配置

**缺省值 [范围]**
No [ Recovery; No ]

这些值如下所示:

- No，表示不保留这些日志。
- Recovery，指示保留这些日志，并可用于正向恢复。

如果 **logretain** 设置为 Recovery 或 **userexit** 设置为 Yes，那么将保留活动日志文件，并且这些文件将成为联机归档日志文件以用于前滚恢复。这称为日志保留日志记录。

在 **logretain** 设置为 Recovery 和/或 **userexit** 设置为 Yes 之后，必须对该数据库进行完全备份。此状态由 **backup_pending** 标志参数指示。

**注:**

**logarchmeth1** 和 **logretain** 都将启用前滚恢复。但是，一次只应对数据库启用一种方法。

如果使用 **logarchmeth1**，请不要设置 **logretain** 和 **userexit** 配置参数。如果 **logretain** 配置参数设置为 Recovery，那么 **logarchmeth1** 的值将自动设置为 **logretain**。

建议使用 **logarchmeth1**（和 **logarchmeth2**）而不是 **logretain** 和 **userexit** 来激活归档记录和前滚恢复。**logretain** 和 **userexit** 选项被保留并支持尚未迁移至 **logarchmeth1** 的用户。

## logsecond - 辅助日志文件数

此参数指定（仅需要时）创建并用于恢复日志文件的辅助日志文件的数目。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
2 [-1; 0 – 254 ]

**计量单位**
计数器

**分配时间**
当 *logprimary* 不够时需要（查看以下详细信息）

**释放时间**
随着时间推移当数据库管理器确定不再需要它们时。

当主日志文件已满时，可按需要一次分配一个辅助日志文件（大小为 *logfilsiz*），最多可分配此参数控制的最大数目。如果需要的辅助日志文件数超过此参数所允许的数，那么将返回错误代码至该应用程序，并将关闭该数据库。

如果将 *logsecond* 设置为 -1，那么数据库将配置为具有无限活动日志空间。对在此数据库上运行的未完成事务的大小和数目没有限制。如果将 *logsecond* 设置为 -1，那么仍使用 *logprimary* 和 *logfilsiz* 配置参数来指定数据库管理器应在活动日志路径中保留多少个日志文件。如果数据库管理器需要读取日志文件中的日志数据，但该文件不在活动日志路径中，那么数据库管理器应将日志文件从归档检索至活动日志路径。（如果配置了溢出日志路径，那么数据库管理器应将文件检索至该路径。）一旦检索到日志文件，数据库管理器会将此文件高速缓存在活动日志路径中，以便他人从同一文件读取日志数据时可提高速度。数据库管理器将根据需要管理这些日志文件的检索、高速缓存和除去操作。

如果您的日志路径是原始设备，那么必须配置 *overflowlogpath* 配置参数以便将 *logsecond* 设置为 -1。

通过将 *logsecond* 设置为 -1，将对工作单元的大小或并发工作单元的数目没有限制。但是，由于需要从压缩文档检索日志文件，回滚（在保存点级别和在工作单元级别）将可能变得很慢。同样的原因，崩溃恢复也可能变得很慢。数据库管理器会将一条消息写入管理通知日志，以通知您当前活动工作单元集已超过主日志文件数。此指示回滚或崩溃恢复可能会非常慢。

要将 *logsecond* 设置为 -1，必须将 *logarchmeth1* 配置参数设置为除 OFF 或 LOGRETAIN 之外的值。

**建议**：对于定期需要大量日志空间的数据库，使用辅助日志文件。例如，每月运行一次的应用程序需要的日志空间可能会超过由主日志文件提供的日志空间。由于辅助日志文件不需要永久的文件空间，所以辅助日志文件在这种情况下有优势。

## max_log - 每个事务的最大日志

此参数指定是否对一个事务可以消耗的日志空间百分比具有限制以及该限制是多少。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
0 [0 — 100 ]

**计量单位**
百分比

如果该值不为 0，那么此参数指示一个事务可以消耗的主日志空间的百分比。

如果该值设置为 0，那么对单个事务可以消耗的空间（按总的主日志空间的百分比计算）没有限制。这是版本 8 之前的事务的行为。

## maxappls - 最大活动应用程序数

此参数指定可与一个数据库连接（本地和远程）的并发应用程序的最大数目。因为每个与数据库连接的应用程序都导致分配一些专用内存，所以允许大量并发应用程序可能将使用更多内存。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
Automatic [1 - 60000]

**计量单位**
计数器

将 **maxappls** 设置为 automatic 将允许任何数目的已连接应用程序。数据库管理器将动态分配它支持新应用程序所需的资源。

如果不想将此参数设置为 automatic，那么此参数的值必须大于或等于已连接应用程序的总数，再加上可能在完成两阶段落实或回滚进程中同时运行的这些相同应用程序的数目。然后将此总和与在任何时刻可能存在的预期不确定事务数相加。

当应用程序试图与数据库连接，但是已经达到了 **maxappls** 时，会向该应用程序返回一个错误，指示已有最大数目的应用程序与数据库连接。

在分区数据库环境中，这是在一个数据库分区同时活动的应用程序的最大数目。此参数限制对于数据库分区服务器上的数据库分区活动的应用程序数，不管该服务器是否是应用程序的协调程序节点。分区数据库环境中的目录节点要求 **maxappls** 的值应比该参数在其他类型的环境中的值高，因为在分区数据库环境中，每个应用程序都需要与该目录节点连接。

**建议**：增大此参数的值而不降低 **maxlocks** 参数或增大 **locklist** 参数的值，可能导致您达到锁定的数据库限制（**locklist**），而不是应用程序限制，从而产生扩散性锁定升级问题。

在一定程度上，**max_coordagents** 也控制应用程序的最大数目。如果有可用的连接（**maxappls**）以及可用的协调代理程序（**max_coordagents**），那么应用程序只能与该数据库连接。

## maxfilop - 每个应用程序打开的最大数据库文件数

此参数确定可以为每个数据库打开的文件句柄的最大数目。

**配置类型**

数据库

**参数类型**

可联机配置

**传播类** 事务边界

**缺省值 [范围]**

**AIX、Sun、HP 和 Linux 64 位**

61 440 [64 - 61 440]

**Linux 32 位**

30 720 [64 - 30 720]

**Windows 32 位**

32 768 [64 - 32 768]

**Windows 64 位**

65 335 [64 - 65 335]

**计量单位**

计数器

如果打开一个文件导致超过此值，那么关闭此数据库正在使用的一些文件。如果 *maxfilop* 太小，打开和关闭文件的开销将变得极大，并且可能会降低性能。

在数据库管理器与操作系统交互时，将 SMS 表空间和 DMS 表空间文件容器作为文件来处理，且需要文件句柄。与一个 DMS 文件表空间所用的容器数相比，SMS 表空间通常要使用更多的文件。因此，如果您使用的是 SMS 表空间，那么将为此参数设置的值应大于为 DMS 文件表空间设置的值。

也可以使用此参数并通过将每个数据库的句柄数限制为一个特定数目，来确保数据库管理器使用的文件句柄总数不超过操作系统限制；实际的限制数将根据同时运行的数据库数不同而有所变化。

## maxlocks -“锁定升级前锁定列表的最大百分比”

此参数定义应用程序所挂起的锁定列表的百分比，必须在数据库管理器执行锁定升级之前填写该列表。

**配置类型**

数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
Automatic [1 - 100 ]

**计量单位**
百分比

锁定升级是用表锁定替换行锁定的过程，以减少列表中的锁定数。当任何一个应用程序所挂起的锁定数达到总锁定列表大小的此百分比时，将发生该应用程序所挂起锁定的锁定升级。当锁定列表空间用尽时也会发生锁定升级。

数据库管理器通过浏览锁定列表以找到应用程序并查找有最多行锁定的表，来确定要升级的锁定。如果用单个表锁定替换这些锁定之后，不再超过 **maxlocks** 值，那么锁定升级将停止。否则，将继续升级，直到保持的锁定列表的百分比低于 **maxlocks** 的值。**maxlocks** 参数乘以 **maxappls** 参数不能小于 100。

当此参数设置为 AUTOMATIC 时，就启用了自调整功能。这允许内存调整器根据工作负载要求变化动态地调整此参数控制的内存区大小。由于内存调整器在不同内存使用者之间交换内存资源，所以，必须至少有两个内存使用者启用自调整功能才能使自调整功能有效。

**locklist** 值是与 **maxlocks** 参数一起调整的，因此，如果禁用 **locklist** 参数自调整功能，也将自动禁用 **maxlocks** 参数自调整功能。如果启用 **locklist** 参数自调整功能，也将自动启用 **maxlocks** 参数自调整功能。

仅当启用了数据库的自调整内存功能（**self_tuning_mem** 配置参数设置为 ON）时，才会自动调整此配置参数。

在 32 位平台上，每个锁需要 48 或 96 字节的锁定列表，这取决于是否对该对象挂起了其他锁:

- 对于没有挂起其他锁的对象，挂起一个锁需要 96 字节
- 对于已经挂起了锁的对象，记录一个锁需要 48 字节。

在 64 位平台（HP-UX 除外）上，每个锁需要 64 或 128 字节的锁定列表，这取决于是否对该对象挂起了其他锁。

- 对于没有挂起其他锁的对象，挂起一个锁需要 128 字节。
- 对于存在一个挂起的锁的对象，记录一个锁需要 64 字节。

在 64 位 HP-UX 上，每个锁需要 80 或 160 字节的锁定列表，这取决于是否对该对象挂起了其他锁。

**建议**: 下列公式允许您设置 **maxlocks**，以允许应用程序保留两次锁定的平均数:

```
maxlocks = 2 * 100 / maxappls
```

其中 2 用来完成两次平均，而 100 表示允许的最大百分比值。如果您仅有几个并发运行的应用程序，那么可以将下列公式当作第一个公式的替代公式:

```
maxlocks = 2 * 100 / (并发运行的应用程序的
平均数目)
```

设置 **maxlocks** 时的其中一项注意事项是将它与锁定列表（**locklist**）的大小配合使用。发生锁定升级之前，应用程序保留的锁定数的实际限制是:

- `maxlocks * locklist * 4096 /(100 * 48)`（在 32 位系统上）
- `maxlocks * locklist * 4096 /(100 * 80)`（在 64 位系统 HP-UX 环境中）
- `maxlocks * locklist * 4096 /(100 * 64)`（在其他 64 位系统上）

其中 `4096` 是页中的字节数，`100` 是 **maxlocks** 所允许的最大百分比值，`48` 是 32 位系统上每个锁的字节数，`80` 是 HP-UX 64 位系统上每个锁的字节数，而 `64` 是其他 64 位系统上每个锁的字节数。如果您知道其中一个应用程序需要 1000 个锁，并且您不希望发生锁定升级，那么应为该公式中的 **maxlocks** 和 **locklist** 选择值，以便结果大于 1000。对 **maxlocks** 使用 10 并且对 **locklist** 使用 100，该公式将产生多于所需的 1000 个锁。

如果将 **maxlocks** 设置的太低，那么当对其他并发应用程序仍然有足够的锁定空间时也会发生锁定升级。如果将 **maxlocks** 设置得太高，那么几个应用程序就可能消耗大多数锁定空间，而其他应用程序将必须执行锁定升级。在此情况下需要锁定升级会导致较差的并行性。

可使用数据库系统监视器帮助您跟踪和调整此配置参数。

## min_dec_div_3 - 十进制除法，小数位为 3 的

此参数提供了一种快速方法来更改 SQL 中十进制除法的小数位计算结果。

**配置类型**
数据库

**参数类型**
可配置

**缺省值 [范围]**
No [Yes, No ]

*min_dec_div_3* 数据库配置参数会更改十进制除法算术运算的结果小数位。它可以设置为“Yes”或“No”。*min_dec_div_3* 的缺省值为“No”。如果值为“No”，那么小数位计算为 31-p+s-s’。如果设置为“Yes”，那么小数位计算为 MAX(3, 31-p+s-s’)。这会导致十进制小数部分始终至少具有 3 位的小数位。精度始终为 31。

更改此数据库配置参数可能会导致更改现有数据库的应用程序。当更改此数据库配置参数会影响十进制小数部分的结果小数位时，就可能发生此情况。下面列示的是可能会影响应用程序的一些可能的情况。在使用现有数据库的数据库服务器上更改 *min_dec_div_3* 之前应该考虑这些情景说明。

- 如果更改其中一个视图列的结果小数位，那么对于在某一环境中用一个设置定义的视图，当更改数据库配置参数之后引用该视图时，该视图可能失败，返回 SQLCODE -344。消息 SQL0344N 指的是递归公共表表达式，但是，如果对象名（第一个标记）是一个视图，那么您将需要删除此视图并再次创建它，以避免此错误。
- 在隐式或显式重新绑定静态程序包之后，静态程序包才会更改行为。例如，将值从 NO 更改为 YES 之后，在执行重新绑定之后，才会在结果中包括附加的小数位。对于任何已更改的静态程序包，显式 REBIND 命令可以用来强制重新绑定。
- 涉及十进制小数部分的检查约束可能会限制先前已接受的某些值。这种行现在违犯约束，但在更新涉及检查约束行的其中一列或处理带有 IMMEDIATE CHECKED 选

项的 SET INTEGRITY 语句之前检测不到。要强制检查这样的约束，执行 ALTER TABLE 语句以删除检查约束，然后执行 ALTER TABLE 语句来再次添加该约束。

**注**: *min_dec_div_3* 还有下列限制:

1. 命令 GET DB CFG FOR DBNAME 将不显示 *min_dec_div_3* 设置。确定当前设置的最好方法是观察十进制小数结果的副作用。例如，考虑如下语句:

   ```
   VALUES (DEC(1,31,0)/DEC(1,31,5))
   ```

   如果此语句返回 SQL 代码 SQL0419N，那么该数据库不具有 *min_dec_div_3* 支持，或者已将它设置为“No”。如果该语句返回 1.000，那么 *min_dec_div_3* 设置为“Yes”。
2. 当运行下列命令时，在配置关键字列表中不会出现 *min_dec_div_3*: ? UPDATE DB CFG

## mincommit - 要分组的落实数

此参数允许您延迟将日志记录写入磁盘，直到执行了最小数目的落实为止，这样有助于减少与写入日志记录相关的数据库管理器开销。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
1 [ 1 – 25 ]

**计量单位**
计数器

如果您有多个应用程序正在对数据库运行，且在很短的时间范围内这些应用程序请求了许多落实，那么此延迟可提高性能。

仅当此参数的值大于 1 且连接至该数据库的应用程序的数目大于或等于此参数的值时，才会发生这种落实组合。当正在执行落实组合，将保留应用程序落实请求直至经过一秒的时间或落实请求的数目等于此参数的值。

只应该稍微增大此参数，例如，增大一 (1)。还应该使用多用户测试来验证增大此参数的值是否能获得期望的结果。

对为此参数指定的值的更改立即生效；您不必等到所有应用程序与该数据库断开连接。

**建议**: 如果多个读/写应用程序在一般情况下请求并发数据库落实，此参数将从缺省值增大。这将导致更有效的日志记录文件 I/O，因为日志记录文件 I/O 进行的频率减小，每次进行日志记录文件 I/O 时能够写入更多的日志记录。

也可以对每秒事务数采样并调整此参数以调整每秒事务数的峰值（或该峰值的较大百分比）。调整峰值活动会使事务高峰期写入日志记录的开销减至最小。

如果增大 *mincommit*，也可能需要增大 *logbufsz* 参数，以避免在这些事务高峰期让已满的日志缓冲区强制写入。在此情况下，*logbufsz* 应该等于:

```
mincommit * (事务使用的日志空间的平均值)
```

可以使用数据库系统监视器来帮助您以下列方式调整此参数:

- 计算每秒的峰值事务数:

  通过抽取典型的一天的监视器样本，您可确定您的事务高峰期。可通过添加下列监视元素来计算总事务数:

  – *commit_sql_stmts* （*尝试的落实语句数*）
  – *rollback_sql_stmts* （*尝试的回滚语句数*）

  使用此信息和可用的时间戳记，就可以计算出每秒事务数。

- 计算每个事务使用的日志空间:

  通过对一段时间和大量事务使用采样技术，您可计算与下列监视元素一起使用的平均日志空间:

  – *log_space_used*（*使用的工作单元日志空间*）

## mirrorlogpath - 镜像日志路径

此参数允许您为镜像日志路径指定最多 242 个字节的字符串。该字符串必须指向路径名，并且它必须为标准路径名，而不是相对路径名。

**配置类型**
数据库

**参数类型**
可配置

**缺省值 [范围]**
Null [任何有效路径或设备]

**注:** 在单分区或多分区 DB2 ESE 环境中，会将节点号自动追加到路径后面。这样做是为了维护多逻辑节点配置中路径的唯一性。

如果配置了 *mirrorlogpath*，那么 DB2 将在日志路径和镜像日志路径中创建活动日志文件。会将所有日志数据写入这两个路径。镜像日志路径有一组相同的活动日志文件，因此如果存在破坏其中一条路径上的活动日志文件的磁盘错误或人为错误，数据库仍能起作用。

如果更改镜像日志路径，那么旧的镜像日志路径中可能有日志文件。可能还没有归档这些日志文件，所以您可能需要手动归档这些日志文件。并且，如果正对此数据库运行复制，那么复制可能仍需要日志路径更改之前的日志文件。如果在将“启用用户出口”（*userexit*）数据库配置参数设置为 Yes 的情况下配置数据库，并且如果 DB2 自动对所有日志文件进行归档或由您手动归档，那么 DB2 将能够检索日志文件以完成复制过程。否则，您可以将文件从旧的镜像日志路径复制到新的镜像日志路径中。

如果 *logpath* 或 *newlogpath* 将原始设备指定为存储日志文件的位置，那么就像 *mirrorlogpath* 指示的那样，不允许镜像日志记录。如果 *logpath* 或 *newlogpath* 将文件路径指定为存储日志文件的位置，那么允许镜像日志记录，并且 *mirrorlogpath* 还必须指定文件路径。

**建议:** 就像日志文件一样，镜像日志文件应在不是高 I/O 的物理磁盘上。

强烈建议此路径在与主日志路径不同的设备上。

可以使用数据库系统监视器来跟踪与数据库日志记录相关的 I/O 数。

下列数据元素返回与数据库日志记录相关的 I/O 活动数。您可以使用操作系统监视工具来收集关于其他磁盘 I/O 活动的信息，然后比较两种类型的 I/O 活动。

- *log_reads*（读取的日志页数）
- *log_writes*（写入的日志页数）。

## multipage_alloc - 启用多页文件分配

多页文件分配用来提高插入性能。它只适用于 SMS 表空间。如果启用多页文件分配，那么影响所有 SMS 表空间: 不可能对各个 SMS 表空间进行选择。

**配置类型**

数据库

**参数类型**

参考

该参数的缺省值为 Yes: 启用了多页文件分配。

创建数据库之后，不能将此参数设置为 No。一旦启用了多页文件分配就不能再将其禁用。可以使用 db2empfa 工具来对当前禁用了多页文件分配的数据库启用多页文件分配。

## newlogpath - 更改数据库日志路径

此参数允许您指定最多 242 个字节的字符串，以更改存储日志文件的位置。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

Null [任何有效路径或设备]

该字符串可以指向路径名或原始设备。请注意，从 DB2 V 9 起，就不推荐使用原始设备来进行数据库日志记录。除了使用原始日志之外，您可使用直接输入/输出 (DIO) 或并发输入/输出 (CIO)。

如果字符串指向路径名，那么它必须是标准路径名，而不能是相对路径名。

在单分区或多分区 DB2 ESE 环境中，会将节点号自动追加到路径后面。这样做是为了维护多逻辑节点配置中路径的唯一性。

如果想要使用复制，并且您的日志路径是原始设备，那么必须配置 *overflowlogpath* 配置参数。

要指定设备，指定操作系统标识成设备的字符串。例如:

- 在 Windows 上，指定 `\\.\d:` 或 `\\.\PhysicalDisk5`。

  **注**: 您必须拥有安装了 Service Pack 3 的 Windows V4.0 或更高版本，才能够将日志写入设备。
- 在 Linux 和 UNIX 平台上，指定 `/dev/rdblog8`。

**注:** 只能在 AIX、Windows 2000、Windows、Solaris、HP-UX 和 Linux 平台上指定设备。

在下列两种情况发生之前，新设置值不会成为 *logpath* 的值:

- 数据库处于一致状态，如 *database_consistent* 参数所指示。
- 所有应用程序将与数据库断开连接

当建立与该数据库的第一个新连接时，数据库管理器将这些日志移至由 *logpath* 指定的新位置。

旧日志路径中可能会有日志文件。这些日志文件可能尚未归档。您可能需要手动归档这些日志文件。并且，如果正对此数据库运行复制，那么复制可能仍需要日志路径更改之前的日志文件。如果在将“启用用户出口”(*userexit*)数据库配置参数设置为 Yes 的情况下配置数据库，并且如果 DB2 自动对所有日志文件进行归档或由您手动归档，那么 DB2 将能够检索日志文件以完成复制过程。否则，可以将这些文件从旧日志路径复制至新日志路径。

如果 *logpath* 或 *newlogpath* 将原始设备指定为存储日志文件的位置，那么就像 *mirrorlogpath* 指示的那样，不允许镜像日志记录。如果 *logpath* 或 *newlogpath* 将文件路径指定为存储日志文件的位置，那么允许镜像日志记录，并且 *mirrorlogpath* 还必须指定文件路径。

**建议:** 理想情况是，这些日志文件将位于**没有**大量 I/O 的物理磁盘上。例如，避免将日志与操作系统或大容量数据库放在同一磁盘上。这将提高日志记录活动的效率并使其开销（例如，等待 I/O）最小。

可以使用数据库系统监视器来跟踪与数据库日志记录相关的 I/O 数。

监视元素 *log_reads*（读取的日志页的数目）和 *log_writes*（写入的日志页的数目）返回与数据库日志记录相关的 I/O 活动数。您可以使用操作系统监视工具来收集关于其他磁盘 I/O 活动的信息，然后比较两种类型的 I/O 活动。

不要将共享网络或本地文件系统用作 DB2 高可用性灾难恢复 (HADR) 数据库对中的主数据库和备用数据库的日志路径。主数据库和备用数据库都有事务日志副本 - 主数据库将日志交付给备用数据库。如果主数据库和备用数据库的日志路径都指向同一个物理位置，那么主数据库和备用数据库会将相同的物理文件用于它们各自的日志副本。如果数据库管理器检测到共享日志路径，那么它就会返回错误。

## num_db_backups - 数据库备份数

此参数指定为一个数据库保留的数据库备份的数目。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 事务边界

**缺省值 [范围]**
12 [1 - 32 767]

当达到指定的备份数时，会在恢复历史记录文件中将旧备份标记为到期。与到期的数据库备份相关的表空间备份和装入副本备份的恢复历史记录文件条目也标记为到期。当备份标记为到期时，可从存储物理备份的地方（例如，磁盘、磁带和 TSM）将它们除去。下一个数据库备份将从恢复历史记录文件中修剪到期的条目。

应将 *rec_his_retentn* 配置参数设置为与 *num_db_backups* 的值兼容的值。例如，如果 *num_db_backups* 设置为一个较大的值，那么 *rec_his_retentn* 的值应当足够大，以支持设置为 *num_db_backups* 的备份数。

## num_freqvalues - 保留的高频值数目

此参数允许您指定在 RUNSTATS 命令中使用 WITH DISTRIBUTION 选项时收集的“最高频值”的数目。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
10 [0 - 32 767 ]

**计量单位**
计数器

增大此参数的值将会增加在收集统计信息时所使用的统计信息堆（*stat_heap_sz*）的数量

“最高频值”统计信息帮助优化器了解列中数据值的分布。较高的值可使查询优化器得到更多信息，但是需要额外的目录空间。当指定为 0 时，即使您请求收集分布统计信息，也不保留任何高频值统计信息。

还可以使用 NUM_FREQVALUES 选项在表或列级别指定作为 RUNSTATS 命令的一部分保留的高频值数目。如果未指定任何值，那么将使用 *num_freqvalues* 配置参数值。通过 RUNSTATS 命令更改保留的高频值数目比通过使用 *num_freqvalues* 数据库配置参数进行更改来得容易。

对于非均匀分布数据上的一些谓词（=、< 或 >），更新此参数能帮助优化器获得更好的选择性估计。选择性计算越精确，选择的访问方案就越有效。

在更改此参数的值之后，您需要:

- 再次运行 RUNSTATS 命令以收集有关已更改的高频值数目的统计信息。
- 重新绑定任何包含静态 SQL 或 XQuery 语句的程序包。

使用 RUNSTATS 时，可在表级别和列级别限制收集的高频值的数目。这允许您通过减少列的分布统计信息来优化目录中占用的空间，在此空间中不能使用这些目录，而这些目录仍然使用关键列的信息。

**建议**：为了更新此参数，您应确定通常具有选择谓词的最重要列（在最重要的表中）中的非均匀度。使用提供每个值在一列中出现次数的有序排序的 SQL SELECT 语句就可实现这一点。不应考虑均匀分布的、唯一的、长的或 LOB 列。此参数合理的实际值范围是从 10 至 100。

注意：收集高频值统计信息的过程需要大量的 CPU 和内存（*stat_heap_sz*）资源。

## num_iocleaners - 异步页清除程序的数目

此参数允许您指定数据库的异步页清除程序数。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

Automatic [0 – 255 ]

**计量单位**

计数器

在数据库代理程序需要缓冲池的空间之前，这些页清除程序将更改的页从缓冲池写入磁盘。因此，数据库代理程序应该不必等待写出已更改的页，它们也能使用缓冲池中的空间。这提高了数据库应用程序的整体性能。

如果您将该参数设置为零（0），那么未启动页清除程序，因此，数据库代理程序将执行将所有页从缓冲池写入磁盘的操作。此参数对存储在许多物理存储设备上的数据库的性能有重大影响，因为在这种情况下很可能其中一个设备将是空闲的。如果未配置页清除程序，那么应用程序可能会遇到周期性日志已满的情况。

如果将此参数设置为 AUTOMATIC，那么启动的页清除程序数取决于当前机器上配置的 CPU 数以及分区数据库环境中的本地逻辑数据库分区数。当此参数设置为 AUTOMATIC 时，始终至少启动一个页清除程序。

当此参数设置为 AUTOMATIC 时，要启动的页清除程序数将使用以下公式计算:

```
number of page cleaners = max( ceil( # CPUs / # local logical DPs ) – 1, 1 )
```

此公式确保在逻辑数据库分区之间均匀地分布页清除程序数，并且页清除程序数不多于 CPU 数。

如果一个数据库的应用程序主要由更新数据的事务组成，那么增加清除程序的数目将会提高运行速度。增加页清除程序也将减少从软故障（例如，断电）恢复的时间，因为磁盘上数据库的内容在任何给定的时间都是最新的。

**建议**：当设置此参数的值时，请考虑下列因素:

- 应用程序类型
  - 如果它是一个不会有更新的只查询数据库，那么将此参数设置为零（0）。例外情况是查询工作负载导致创建许多 TEMP 表（您可以使用说明实用程序来确定这点）。
  - 如果事务是对数据库运行的，那么将此参数的值设置在 1 与用于数据库的物理存储设备数之间。

- 工作负载

  具有较高更新事务比率的环境可能需要配置更多页清除程序。
- 缓冲池大小

  含有大缓冲池的环境可能也需要配置更多的页清除程序。

可以使用数据库系统监视器来帮助调整此配置参数，该数据库系统监视器使用事件监视器收集的有关缓冲池写入活动的信息:

- 如果同时满足下面两个条件，那么可以减小该参数:
  - *pool_data_writes* 大约等于 *pool_async_data_writes*
  - *pool_index_writes* 大约等于 *pool_async_index_writes*。
- 如果下列任何一种情况为真，那么应增大该参数:
  - *pool_data_writes* 比 *pool_async_data_writes* 大很多
  - *pool_index_writes* 比 *pool_async_index_writes* 大很多

## num_ioservers -“I/O 服务器数”

此参数指定用于数据库的 I/O 服务器的数目。无论任何时候，对一个数据库数据库执行的预取和实用程序的 I/O 数都不能超过此数目。

**配置类型**
数据库

**参数类型**
可配置

**缺省值 [范围]**
Automatic [1 – 255 ]

**计量单位**
计数器

**分配时间**
当应用程序连接至数据库时

**释放时间**
当应用程序与数据库断开连接时

用 I/O 服务器（又称为“预取程序”）代表数据库代理程序从而通过实用程序（例如，backup 实用程序和 restore 实用程序）执行预取 I/O 和异步 I/O。I/O 服务器在执行它所启动的 I/O 操作期间等待。非预取 I/O 直接由数据库代理程序调度，因此，它不受 *num_ioservers* 约束。

如果此参数设置为 AUTOMATIC，那么启动的预取程序数将基于当前数据库分区中的表空间并行性设置。

当此参数设置为 AUTOMATIC 时，将在激活数据库时根据以下公式计算所要启动的预取程序数:

```
number of prefetchers = max( max over all table spaces( parallelism setting ), 3 )
```

其中并行性设置由 *DB2_PARALLEL_IO* 系统环境变量控制。

**建议**: 为了最大限度地利用系统中的所有 I/O 设备，理想的值通常是比该数据库所在的物理设备的数目多 1 或 2 个。最好配置额外的 I/O 服务器，因为每个 I/O 服务器都有关联的最小开销，而且任何未使用的 I/O 服务器都将保持为空闲。

## num_log_span - 跨越的日志数

此参数指定是否对一个事务可以跨越多少个日志文件具有限制以及该限制是多少。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
0 [0 - 65 535 ]

**计量单位**
计数器

如果该值不为 0，那么此参数指示允许一个活动事务跨越的活动日志文件数。

如果该值设置为 0，那么对单个事务可以跨越的日志文件数没有限制。这是版本 8 之前的事务的行为。

## num_quantiles - 列的分位数

此参数控制在 RUNSTATS 命令中指定 WITH DISTRIBUTION 选项时将收集的分位数的数目。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
20 [0 - 32 767]

**计量单位**
计数器

增大此参数的值将会增加在收集统计信息时所使用的统计信息堆（*stat_heap_sz*）的数量

“分位数”统计信息帮助优化器了解列中数据值的分布。较高的值可使查询优化器得到更多信息，但是需要额外的目录空间。指定 0 或 1 时，不保留分位数统计信息，即使您请求收集分布统计信息。

还可以使用 NUM_QUANTILES 选项在表或列级别指定作为 RUNSTATS 命令的一部分收集的分位数的数目。如果未指定任何值，那么将使用 *num_quantiles* 配置参数值。通过 RUNSTATS 命令更改将收集的分位数的数目比通过使用 *num_quantiles* 数据库配置参数进行更改容易。

对于非均匀分布数据上的范围谓词，更新此参数能帮助获得更好的选择性估计。在其他优化器决策中，此信息对于选择索引扫描还是选择表扫描具有很大的影响。（访问频繁出现的值的范围时使用表扫描更为有效，而对不频繁出现的值的范围则使用索引扫描更为有效。）

在更改此参数的值之后，您需要:

- 再次运行 RUNSTATS 命令以收集有关已更改的高频值数目的统计信息。
- 重新绑定任何包含静态 SQL 或 XQuery 语句的程序包。

使用 RUNSTATS 时，可在表级别和列级别限制收集的分位数的数目。这允许您通过减少列的分布统计信息来优化目录中占用的空间，在此空间中不能使用这些目录，而这些目录仍然使用关键列的信息。

**建议**: 此参数的缺省值保证，对于任何单方范围谓词（>、>=、< 或 <=）的最大估计错误大约为 2.5%，而对于任何 BETWEEN 谓词的最大错误则为 5%。计算分位数数目的近似值的简单方法是:

- 确定在估计任何范围查询的行数时可允许的最大错误百分比，即 P。
- 如果大多数谓词为 BETWEEN 谓词，那么分位数的数目应大约为 100/P，如果大多数的谓词是其他类型的范围谓词（<、<=、> 或 >=），那么分位数的数目大约为 50/P。

例如，25 个分位数导致的最大估计错误对于 BETWEEN 谓词应为 4%，而对于“>”谓词则应为 2%。此参数合理的实际值范围为 10 至 50。

## numarchretry - 出错时重试次数

此参数指定在 DB2 尝试将日志文件归档至故障转移目录之前，它尝试将日志文件归档至主归档目录或辅助归档目录的次数。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

5 [0 - 65 535 ]

仅当设置了 *failarchpath* 数据库配置参数时才使用此参数。如果未设置 *numarchretry*，那么 DB2 将不断重试归档至主要日志路径或辅助日志路径。

## numsegs - 缺省 SMS 容器数

在版本 9.5 中不推荐使用此参数，但是版本 9.5 之前的数据服务器和客户机仍然使用此参数。DB2 V9.5 数据库管理器将忽略对此配置参数指定的任何值。

**注:** 下列信息仅适用于版本 9.5 之前的数据服务器和客户机。

**配置类型**
数据库

**参数类型**
参考

**计量单位**
计数器

此参数指示将在缺省表空间中创建的容器的数目。它还显示在创建数据库时使用的信息，无论此参数在 CREATE DATABASE 命令中是显式还是隐式指定的。

此参数仅适用于 SMS 表空间；CREATE TABLESPACE 语句不以任何方式使用此参数。

## number_compat -"数字兼容性"数据库配置参数

此参数指示与 NUMBER 数据类型相关联的兼容性语义是否应用于所连接的数据库。

**配置类型**
数据库

**参数类型**
参考

值在数据库创建时确定，并且基于用于 NUMBER 支持的 DB2_COMPATIBILITY_VECTOR 注册表变量的设置。值不可更改。

## overflowlogpath - 溢出日志路径

此参数指定一个位置，DB2 数据库将在该位置中查找前滚操作所需的日志文件，并将从归档中检索到的活动日志文件存储在该位置。它还提供了一个位置，用于查找和存储在使用 db2ReadLog API 时所需要的日志文件。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
NULL [任何有效路径]

此参数可以用于几种函数，这取决于日志记录需求。

- 此参数允许您指定一个位置，供 DB2 数据库查找前滚操作所需要的日志文件。此参数与 ROLLFORWARD 命令的 **OVERFLOW LOG PATH** 选项相似。您只需设置此配置参数一次，而不必在每个 ROLLFORWARD 命令上都指定 **OVERFLOW LOG PATH**。但是，如果您同时使用这两种方法，那么对于该特定前滚操作，**OVERFLOW LOG PATH** 选项将覆盖 **overflowlogpath** 配置参数。
- 如果将 **logsecond** 设置为 -1，那么 **overflowlogpath** 允许您为 DB2 指定一个目录用来存储从归档中检索到的活动日志文件。（如果活动日志文件不再存在于活动日志路径中，那么必须检索它们以用于回滚操作）。如果没有 **overflowlogpath**，那么 DB2 数据库将把日志文件检索到活动日志路径中。使用 **overflowlogpath** 时，

允许您为 DB2 数据库提供附加资源以存储所检索到的日志文件。好处包括将 I/O 成本分布到不同的磁盘上，以及允许将更多的日志文件存储在活动日志路径中。

- 例如，如果您需要使用 db2ReadLog API（在 DB2 版本 8 之前，db2ReadLog 称为 sqlurlog）来进行复制，那么 **overflowlogpath** 允许您指定一个位置供 DB2 数据库用来搜索此 API 需要的日志文件。如果未找到日志文件（在活动日志路径或溢出日志路径中）且数据库配置为启用 **userexit**，那么 DB2 将检索日志文件。**overflowlogpath** 还允许您指定一个目录供 DB2 数据库用来存储所检索到的日志文件。好处包括降低活动日志路径上的 I/O 成本以及允许将更多的日志文件存储在活动日志路径中。
- 如果您已经为活动日志路径配置了原始设备，那么如果要将 **logsecond** 设置为 -1，或者想要使用 db2ReadLog API，就必须配置 **overflowlogpath**。

要设置 **overflowlogpath**，指定一个最长 242 个字节的字符串。该字符串必须指向路径名，并且它必须为标准路径名，而不是相对路径名。该路径名必须是目录，而不是原始设备。

**注**：在分区数据库环境中，会自动将节点号追加至路径后面。这样做是为了维护多逻辑节点配置中路径的唯一性。

## pagesize - 数据库缺省页大小

此参数包含在创建数据库时用作缺省页大小的值。可能的值包括：4096、8192、16384 和 32768。在当该数据库中创建缓冲池或表空间时，应用同样的缺省页大小。

**配置类型**

数据库

**参数类型**

参考

## pckcachesz - 程序包高速缓存大小

此参数是在数据库共享内存之外分配的，并且用于高速缓存数据库上的静态和动态 SQL 和 XQuery 语句的部分。

**配置类型**

数据库

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

**32 位平台**

Automatic [-1, 32 - 128 000]

**64 位平台**

Automatic [-1, 32 - 524 288]

**计量单位**

页 (4 KB)

**分配时间**

当初始化数据库时

**释放时间**

当数据库关闭时

在一个分区数据库系统中，每个数据库分区都有一个程序包高速缓存。

高速缓存程序包使数据库管理器在重新装入程序包时可以不访问系统目录；或者对于动态 SQL 或 XQuery 语句，可以免去编译这一步，从而减少其内部开销。将这些段保存在程序包高速缓存中，直到发生下列其中一种情况:

- 数据库关闭
- 程序包或动态 SQL 或 XQuery 语句无效
- 高速缓存空间用完。

静态或动态 SQL 或 XQuery 语句节的高速缓存可提供性能，尤其是在与数据库连接的应用程序多次使用同一个语句时。这在事务处理环境中特别重要。

当此参数设置为 AUTOMATIC 时，就启用了自调整功能。当 *self_tuning_mem* 设置为 ON 时，内存调整器将在工作负载要求更改时动态调整 *pckcachesz* 控制的内存区的大小。由于内存调整器在不同内存使用者之间交换内存资源，所以，必须至少有两个内存使用者启用自调整功能才能使自调整功能有效。

仅当启用了数据库的自调整内存功能（*self_tuning_mem* 配置参数设置为“ON”）时，才会自动调整此配置参数。

当此参数设置为 -1 时，用来计算页分配的值是为 *maxappls* 配置参数指定的值的 8 倍。例外情况是 *maxappls* 的 8 倍小于 32。在这种情况下，缺省值 -1 将 *pckcachesz* 设置为 32。

**建议**：当调整此参数时，应考虑如果为程序包高速缓存保留的额外内存是为另一目的分配的（如缓冲池或目录高速缓存），它是否会更有效。因此，应在调整此参数时使用基准程序技术。

当最初使用几节，而后只有少数几节反复运行时，调整此参数就特别重要。如果高速缓存太大，那么因保存初始节的副本而浪费内存。

下列监视元素可以帮助您确定是否应调整此配置参数:

- *pkg_cache_lookups*（程序包高速缓存查询数）
- *pkg_cache_inserts*（程序包高速缓存插入数）
- *pkg_cache_size_top*（程序包高速缓存高水位标记）
- *pkg_cache_num_overflows*（程序包高速缓存溢出）

**注**：程序包高速缓存是工作高速缓存，所以不能将此参数设置为零。此高速缓存中必须分配有足够的内存以保存当前执行的 SQL 或 XQuery 语句的所有节。如果分配的空间比当前需要的空间多，那么各节被高速缓存。下一次需要这些部分时，只需执行它们而不必将其装入或进行编译。

由 *pckcachese* 参数指定的限制是软限制。如果数据库共享集合中还有可用的内存，如果有必要，可以超过该限制。可使用 *pkg_cache_size_top* 监视元素确定程序包高速缓存达到的最大值，并用 *pkg_cache_num_overflows* 监视元素确定超过了由 *pckcachesz* 参数指定的限制的多少倍。

## priv_mem_thresh - 专用内存阈值

在版本 9.5 中不推荐使用此参数，但是版本 9.5 之前的数据服务器和客户机仍然使用此参数。DB2 V9.5 数据库管理器将忽略对此配置参数指定的任何值。

**注**: 下列信息仅适用于版本 9.5 之前的数据服务器和客户机。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可配置

**缺省值 [范围]**

20000 [-1; 32 - 112000]

**计量单位**

页 (4 KB)

此参数用于确定未用过的代理程序专用内存量，这些内存将处于分配状态，准备供启动的新代理程序使用。它不适用于 Linux 和 UNIX 平台。

值 -1 将导致此参数使用 *min_priv_mem* 参数的值。

**建议**: 当设置此参数时，您应考虑客户机连接/断开连接模式，以及同一台机器上其他进程的内存需求。

如果仅在一个短暂的时期内有很多客户机同时与数据库连接，那么高阈值将防止未使用的内存被取消落实，从而被其他进程使用。这种情况将导致内存管理混乱，从而会影响其他需要内存的进程。

如果并发客户机数围绕一个固定值频繁变动，那么高阈值将帮助确保内存可用于客户机进程并减少分配内存和释放内存的开销。

## rec_his_retentn - 恢复历史记录保留期

此参数指定将保留有关备份的历史记录信息的天数。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

366 [-1; 0 - 30 000]

**计量单位**

天数

如果不需要恢复历史记录文件来跟踪备份、复原以及装入，那么可将此参数值设置为一个较小的数。

如果 **rec_his_retentn** 设置为 -1 并且 **auto_del_rec_obj** 设置为 OFF，那么指示完全数据库备份（以及与数据库备份相关联的任何表空间备份）的条目数将对应于 **num_db_backups** 数据库配置参数所指定的值。只能使用可用的命令或 API 显式修剪恢复历史记录文件中的其他条目。如果 **rec_his_retentn** 设置为 -1 并且 **auto_del_rec_obj** 设置为 ON，那么不会自动修剪历史记录文件并且不会删除恢复对象。

如果 **rec_his_retentn** 设置为 0 并且 **auto_del_rec_obj** 设置为 OFF，那么将修剪历史记录文件中除最后一次生成的完全备份之外的所有条目。如果 **auto_del_rec_obj** 设置为 ON，那么将根据 **num_db_backups** 数据库配置参数所选择的备份的时间戳记来执行自动修剪历史记录文件和删除恢复对象。

不管保留期多短，将总是保留最近的完全数据库备份及其复原集，除非您使用带有 FORCE 选项的 PRUNE 命令。

## restore_pending - 复原暂挂

此参数说明数据库中是否有 RESTORE PENDING 状态。

**配置类型**
数据库

**参数类型**
*参考*

## restrict_access - 数据库具有受限访问配置参数

此参数指示数据库是否是使用一组受限的缺省操作创建的。也就是说，创建该数据库时，在 CREATE DATABASE 命令中是否指定了 RESTRICTIVE 子句。

**配置类型**
数据库

**参数类型**
*参考*

**YES** 创建此数据库时，在 CREATE DATABASE 命令中使用了 RESTRICTIVE 子句。

**NO** 创建此数据库时，在 CREATE DATABASE 命令中未使用 RESTRICTIVE 子句。

## rollfwd_pending - 前滚暂挂指示器

此参数告知您是否需要执行前滚恢复以及需要执行前滚恢复的位置。

**配置类型**
数据库

**参数类型**
*参考*

此参数可指示下列其中一种状态:

- **DATABASE**，表示此数据库必需前滚恢复过程

- **TABLESPACE**，表示一个或多个表空间需要前滚
- **NO**，表示该数据库是可用的，且不需要前滚恢复。

在可以访问数据库或表空间之前，必须完成恢复（使用 ROLLFORWARD DATABASE）。

## self_tuning_mem - 自调整内存

此参数确定内存调整器是否根据需要在启用了自调整功能的内存使用者之间动态地分配可用内存资源。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**

**单个数据库分区环境**
ON [ ON; OFF]

**多个数据库分区环境**
OFF [ ON; OFF]

在从先前版本迁移的数据库中，*self_tuning_mem* 将设置为 OFF。

由于要在内存使用者之间交换内存，因此必须至少有两个启用了自调整功能的内存使用者才能使内存调整器处于活动状态。当 *self_tuning_mem* 设置为 ON，但启用了自调整功能的内存使用者不足两个时，内存调整器将处于不活动状态。（排序堆内存区例外，无论其他内存使用者是否启用了自调整功能，都可以对排序堆内存区进行调整。）当 *database_memory* 设置为一个数值时，认为启用了自调整功能。

在单数据库分区环境中，缺省情况下，此参数为 ON。在多个数据库分区环境中，缺省情况下，此参数为 OFF。

可以启用自调整功能的内存使用者包括:

- 缓冲池（由 ALTER BUFFERPOOL 和 CREATE BUFFERPOOL 语句的大小参数控制）
- 程序包高速缓存（由 *pckcachesz* 配置参数控制）
- 锁定列表（由 *locklist* 和 *maxlocks* 配置参数控制）
- 排序堆（由 *sheapthres_shr* 和 *sortheap* 配置参数控制）
- 数据库共享内存（由 *database_memory* 配置参数控制）

要查看此参数的当前设置，请使用指定了 SHOW DETAIL 参数的 GET DATABASE CONFIGURATION 命令。对此参数返回的可能设置是:

```
Self Tuning Memory          (SELF_TUNING_MEM) = OFF
Self Tuning Memory          (SELF_TUNING_MEM) = ON (Active)
Self Tuning Memory          (SELF_TUNING_MEM) = ON (Inactive)
Self Tuning Memory          (SELF_TUNING_MEM) = ON
```

下列值指示:

- ON (Active) - 内存调整器当前正在调整系统上的内存
- ON (Inactive) - 虽然参数设置为 ON，但由于启用自调整功能的内存使用者不足两个，因此未执行自调整
- ON（不带 (Active) 或 (Inactive)）- 由未使用 SHOW DETAIL 选项或没有数据库连接的查询产生。

在分区环境中，对于正在运行调整器的数据库分区，*self_tuning_mem* 配置参数将只显示 ON (Active)。在所有其他节点上，*self_tuning_mem* 将显示 ON (Inactive)。因此，要确定内存调整器在分区数据库中是否活动，必须检查所有数据库分区上的 *self_tuning_mem* 参数。

如果已从较早版本的 DB2 迁移到 DB2 V9，并且计划使用自调整内存功能，那么应该配置下列运行状况指示器以禁止检查阈值或状态:

- 共享排序内存利用率 - db.sort_shrmem_util
- 溢出排序百分比 - db.spilled_sorts
- 长期共享的排序内存利用率 - db.max_sort_shrmem_util
- 锁定列表利用率 - db.locklist_util
- 锁定升级率 - db.lock_escal_rate
- 程序包高速缓存命中率 - db.pkgcache_hitratio

自调整内存功能的其中一个目标是避免将内存分配给并不立即需要内存的内存使用者。因此，在分配更多内存前，分配给内存使用者的内存的利用率可能会达到 100%。通过禁用这些运行状况指示器，可以避免内存使用者的高内存利用比率不必要地触发警报。

缺省情况下，在 DB2 V9 中创建的实例将禁用这些运行状况指示器。

## seqdetect - 顺序检测标志

此参数控制是否允许数据库管理器在 I/O 活动期间检测有序页读取。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
Yes [ Yes; No ]

数据库管理器可以监视 I/O，并且如果正在进行顺序页读取，那么数据库管理器可以激活 I/O 预取。这种类型的顺序预取称为*顺序检测*。

如果此参数设置为 No，那么仅当数据库管理器知道预取将有用时才会发生预取。例如，表排序、表扫描或列表预取。

**建议**：在大多数情况下，应使用此参数的缺省值。仅当其他调整都不能解决严重的查询性能问题时，才尝试关闭顺序检测。

## sheapthres_shr - 共享排序的排序堆阈值

此参数表示对排序内存使用者每次可使用的数据库共享内存总量的软限制。

**配置类型**

数据库

**适用于** OLAP 函数

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

**32 位平台**

Automatic [250 - 524288]

**64 位平台**

Automatic [250 - 2 147 483 647]

**计量单位**

页 (4 KB)

除排序以外，还有其他排序内存使用者（例如，散列连接、索引 AND 运算、块索引 AND 运算、合并连接和内存表）。当共享排序内存使用者的共享排序总量达到 *sheapthres_shr* 限制时，就会激活内存调节机制，并且将来的共享排序内存使用者请求得到的内存量将少于请求的内存量，但始终多于完成任务所需的最低内存量。一旦达到 *sheapthres_shr* 限制，排序内存使用者的所有共享排序内存请求都将获得完成任务所必需的最低内存量。当活动共享排序内存使用者的共享内存总量达到此限制时，后续排序可能会失败（SQL0955C）。

当数据库管理器配置参数 *sheapthres* 值为 0 时，数据库的所有排序内存使用者都将使用 *sheapthres_shr* 控制的数据库共享内存，而不是使用专用排序内存。

当 *sheapthres_shr* 设置为 AUTOMATIC 时，就启用了自调整功能。这允许内存调整器根据工作负载要求变化动态地调整此参数控制的内存区大小。由于内存调整器在不同内存使用者之间交换内存资源，所以，必须至少有两个内存使用者启用自调整功能才能使自调整功能有效。内存使用者包括 SHEAPTHRES_SHR、PCKCACHESZ、BUFFER POOL（每个缓冲池计数为一个）、LOCKLIST 和 DATABASE_MEMORY。

仅当数据库管理器配置参数 *sheapthres* 设置为 0 时，才允许自动调整 *sheapthres_shr*。

*sortheap* 值是与 *sheapthres_shr* 参数一起调整的，因此，如果禁用 *sortheap* 参数自调整功能，也将自动禁用 *sheapthres_shr* 参数自调整功能。如果启用 *sheapthres_shr* 参数自调整功能，也将自动启用 *sortheap* 参数自调整功能。

仅当启用了数据库的自调整内存功能（*self_tuning_mem* 配置参数设置为“ON”）时，才会自动调整此配置参数。

在联机状态下更新此参数值时，只有更新后发出的新共享排序内存请求才会使用新值。建议您在减小 *sheapthres_shr* 值之前减小 *sortheap* 值，并且在增大 *sortheap* 值之前增大 *sheapthres_shr* 值。

当数据库管理器配置参数 *sheapthres* 大于 0 时，*sheapthres_shr* 只有在两种情况下有意义:

- 当 *intra_parallel* 数据库管理器配置参数设置为 *yes* 时，这是因为当 *intra_parallel* 设置为 *no* 时，没有任何共享排序。
- 当集中器处于打开状态时（即，当 *max_connections* 大于 *max_coordagents* 时），这是因为，如果排序使用在声明时指定了 WITH HOLD 选项的游标，就会为该排序分配共享内存。

## softmax - 恢复范围和软检查点时间间隔

此参数确定软检查点的频率和恢复范围，这将有助于完成崩溃恢复过程。

**配置类型**

数据库

**参数类型**

可配置

**缺省值 [范围]**

100 [ 1 – 100 * *logprimary* ]

**计量单位**

一个主日志文件的大小百分比

此参数用来:

- 影响在一次崩溃（如电源故障）之后需要恢复的日志数。例如，如果使用缺省值，那么数据库管理器将尝试把需要恢复的日志数保持为 1。如果您指定 300 作为此参数的值，那么数据库管理器将尝试把需要恢复的日志数保持为 3。

  要影响进行崩溃恢复所需要的日志数，数据库管理器使用此参数来触发页清除程序，以确保比指定的恢复窗口旧的页都已写入磁盘。
- 确定软检查点的频率。

当由于事件（例如，电源故障）造成数据库故障时，可能会对该数据库作下列更改:

- 尚未落实，但更新了缓冲池中的数据
- 已落实，但尚未从缓冲池写入磁盘
- 已落实并从缓冲池写入磁盘。

当重新启动某个数据库时，将使用日志文件来执行该数据库的崩溃恢复，这确保该数据库处于一致状态（即，所有已落实的事务都应用于该数据库而所有未落实的事务都不应用于该数据库）。

要确定需要将日志文件中的哪些记录应用于该数据库，数据库管理器使用日志控制文件中所记录的信息。（数据库管理器实际上保持了日志控制文件的两个副本，即 SQLOGCTL.LFH.1 和 SQLOGCTL.LFH.2，以便在一个副本被破坏的情况下，数据库管理器还可以使用另一个副本。）定期将这些日志控制文件写入磁盘，并且根据此事件的频率，数据库管理器可能正在应用已落实事务的日志记录，或正在应用那些描述已从缓冲池写入磁盘的更改的日志记录。这些日志记录对数据库没有影响，但应用这些日志记录会将一些开销引入到数据库重新启动进程中。

当日志文件已满时以及在软检查点期间，始终将日志控制文件写入磁盘。可使用此配置参数来触发其他软检查点。

软检查点的定时基于“当前状态”与“记录的状态”之间的差别，该差别以 *logfilsiz* 的百分比给出。“记录的状态”由磁盘上日志控制文件中指示的最旧的有效日志记录确定，而“当前状态”由内存中的日志控制信息确定。（最旧的有效日志记录是恢复进程将读取的第一个日志记录。）如果下列公式计算的值大于或等于此参数的值，将采用该软检查点:

```
( (记录的状态和当前状态之间的空间) / logfilsiz ) *  100
```

**建议**: 您可能想增大或减小此参数的值，这取决于您的可接受的恢复窗口是大于还是小于一个日志文件。降低此参数的值将导致数据库管理器不但更经常地触发页清除程序而且还更频繁地采用软检查点。这些操作可减少需要处理的日志记录数和在崩溃恢复期间处理的冗余日志记录数。

但是注意: 更多的页清除程序触发器和更频繁的软检查点增加了与数据库日志记录相关联的开销，这可能会影响数据库管理器的性能。而且，如果您遇到下列情况，更频繁的软检查点可能不会缩短重新启动一个数据库所需的时间:

- 落实点很少的很长的事务。
- 很大的缓冲池并且未将包含落实事务的页很频繁地写回至磁盘。（注意，使用异步页清除程序可能有助于避免此情况。）

在这两种情况中，保留在内存中的日志控制信息不会频繁更改，因而在将日志控制信息写入磁盘时不存在优势，除非日志控制信息已更改。

## sortheap - 排序堆大小

此参数定义要用于专用排序的专用内存页的最大数目或要用于共享排序的共享内存页的最大数目。

**配置类型**
数据库

**适用于** OLAP 函数

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**

**32 位平台**
Automatic [16 - 524288]

**64 位平台**
Automatic [16 - 4194303]

**计量单位**
页 (4 KB)

**分配时间**
需要执行排序时

**释放时间**
当排序完成时

如果排序为专用排序，那么此参数将影响代理程序专用内存。如果排序为共享排序，那么此参数将影响数据库共享内存。每个排序都有一个独立的排序堆，该排序堆由数

据库管理器根据需要分配。此排序堆是将数据排序的区域。如果由优化器定向，那么将使用优化器提供的信息分配一个比此参数指定的排序堆小的排序堆。

当此参数设置为 AUTOMATIC 并且 **sheapthres** 设置为 0 时，就启用了自调整功能。这允许内存调整器根据工作负载要求变化动态地调整此参数控制的内存区大小。

**sortheap** 值是与 **sheapthres_shr** 参数一起调整的，因此，如果不禁用 **sheapthres_shr** 参数自调功能，便不能禁用 **sortheap** 参数自调功能。如果启用 **sheapthres_shr** 参数自调整功能，也将自动启用 **sortheap** 参数自调整功能。但是，即使 **sheapthres_shr** 参数不是 AUTOMATIC，也可以启用 **sortheap** 参数自调整功能。

仅当数据库管理器配置参数 **sheapthres** 设置为 0 时，才允许自动调整 **sortheap**。

仅当启用了数据库的自调整内存功能（**self_tuning_mem** 配置参数设置为 ON）时，才会自动调整此配置参数。

**建议**：当使用排序堆时，应该考虑下列事项:

- 适当的索引可使排序堆的使用减至最小程度。
- 散列连接缓冲区、块索引 AND 运算、合并连接、内存中的表以及动态位映射（用于索引 AND 运算和星型连接）使用排序堆内存。在使用这些技术时，增大此参数的大小。
- 当需要进行频繁的大型排序时，增大此参数的大小。
- 当增大此参数的值时，应该检查是否还需要调整数据库管理器配置文件中的 **sheapthres** 和 **sheapthres_shr** 参数。
- 排序堆大小由优化器在确定访问路径时使用。在更改此参数之后，应考虑（使用 REBIND 命令）重新绑定应用程序。

更新 **sortheap** 值之后，数据库管理器会立即开始将此新值用于任何当前排序或新排序。

## stat_heap_sz -“统计信息堆大小”

此参数指示使用 RUNSTATS 命令收集统计信息时所用的最大堆大小。

对于版本 9.5，此数据库配置参数的缺省值为 AUTOMATIC，这表示它将根据需要增大，直到达到 *appl_memory* 限制或达到 *instance_memory* 限制。

**配置类型**
数据库

**参数类型**
可联机配置

**缺省值 [范围]**
Automatic [1 096 - 524 288]

**计量单位**
页 (4 KB)

**分配时间**
当启动 RUNSTATS 实用程序时

**释放时间**

当 RUNSTATS 实用程序完成时

**建议**: RUNSTATS 内存需求取决于几个因素。统计信息选项越多，使用的内存也就越多，例如，如果正在收集 LIKE 统计信息或 DETAILED 索引统计信息。收集列统计信息时，收集的列统计信息的数目越多，使用的内存也越多。收集分布统计信息时，收集的频率和/或分位数值越多，使用的内存也越多。建议使用缺省设置 AUTOMATIC。

## stmtheap -“语句堆大小”

此参数指定语句堆的大小，语句堆在编译 SQL 或 XQuery 语句期间用作 SQL 或 XQuery 编译器的工作空间。

**配置类型**

数据库

**参数类型**

可联机配置

**传播类** 语句边界

**缺省值 [范围]**

**对于 32 位平台**

AUTOMATIC [128 - 524288]

- 带有本地和远程客户机的数据库服务器: 缺省值为 AUTOMATIC（底层值为 2048）。
- 还可将该参数仅设置为某个固定值。

**对于 64 位平台**

AUTOMATIC [128 - 524288]

- 带有本地和远程客户机的数据库服务器: 缺省值为 AUTOMATIC（底层值为 8192）。
- 还可将该参数仅设置为某个固定值。

**计量单位**

页 (4 KB)

**分配时间**

对于预编译或绑定期间的每个语句

**释放时间**

当每个语句的预编译或绑定完成时

此区域并不总是处于分配状态，但要对每个处理的 SQL 或 XQuery 语句进行分配和释放。注意: 对于动态 SQL 或 XQuery 语句，将在程序执行期间使用此工作区；而对于静态 SQL 或 XQuery 语句，在绑定进程而不是在程序执行期间使用此工作区。

可使用底层值或固定值将 **STMTHEAP** 参数设置为 AUTOMATIC。该参数设置为 AUTOMATIC 时，底层值会对使用动态连接枚举为单个编译分配的内存量加以限制。如果遇到内存限制，语句编译会使用贪婪连接枚举和不受限制的语句堆重新启动。其仅受到剩余应用程序内存量 (**APPL_MEMORY**)、实例内存 (**INSTANCE_MEMORY**) 或系统内存的限制。如果贪婪连接枚举成功完成，那么将向应用程序返回一个 SQL0437W 警告。如果贪婪连接枚举也遇到内存限制，那么语句预编译失败，且带有 SQL0101N。

例如，db2 update db cfg for SAMPLE using STMTHEAP 8192 AUTOMATIC 将为动态连接枚举设置 8192 *4K (32MB) 的语句堆限制并对贪婪连接枚举不设限制。

当 STMTHEAP 参数设置为固定值时，该限制同时适用于动态和贪婪连接枚举。如果动态连接枚举遇到内存限制，那么将使用同一固定语句堆限制来尝试贪婪连接枚举。在 AUTOMATIC 情况下，类似的警告/错误也适用。

例如，db2 update db cfg for SAMPLE using STMTHEAP 8192 将同时为动态和贪婪连接枚举生成 8192 * 4K (32MB) 的语句堆限制。

如果您的应用程序收到 SQL0437W 警告，且查询的运行时性能不足，那么您可能要考虑增加 stmtheap 值（底层为 AUTOMATIC 的值或固定值），从而确保动态编程连接枚举能够成功。

**注:** 仅在优化类 3 和更高级别（缺省值为 5）进行动态编程联合枚举。

## territory - 数据库地域

此参数显示用于创建数据库的地域。数据库管理器在处理地域相关的数据时会使用 *territory*。

**配置类型**
数据库

**参数类型**
参考

## trackmod - 启用跟踪已修改页

此参数指定数据库管理器是否将跟踪数据库修改，以便 Backup 实用程序可以检测到数据库页的哪些子集必须通过增量备份来检查并可能包括在备份映像中。

**配置类型**
数据库

**参数类型**
可配置

**缺省值 [范围]**
No [Yes, No ]

在将此参数设置为“YES”之后，必须执行完全数据库备份，才能获得可以对其执行增量备份的基线。此外，如果启用此参数并创建了一个表空间，那么必须执行包含该表空间的备份。此备份可以是数据库备份，也可以是表空间备份。在执行备份之后，将允许执行增量备份来包含此表空间。

## tsm_mgmtclass - Tivoli Storage Manager 管理类

Tivoli Storage Manager 管理类确定 TSM 服务器应如何管理正在备份的对象的备份版本。

**配置类型**
数据库

**参数类型**
可配置

**缺省值 [范围]**
Null [任何字符串]

缺省情况是没有 DB2 指定的管理类。

当执行任何 TSM 备份时，在使用数据库配置参数指定的管理类之前，TSM 首先尝试将备份对象绑定到 TSM 客户机选项文件中的包含/排除列表中指定的管理类。如果找不到匹配项，那么将使用 TSM 服务器上指定的缺省 TSM 管理类。然后，TSM 将备份对象重新绑定到数据库配置参数指定的管理类。

因此，缺省管理类以及数据库配置参数指定的管理类必须包含备份副本组，否则备份操作将失败。

## tsm_nodename - Tivoli Storage Manager 节点名

此参数用于覆盖与 Tivoli Storage Manager（TSM）产品相关的节点名的缺省设置。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 语句边界

**缺省值 [范围]**
Null [任何字符串]

在复原从另一个节点备份至 TSM 的数据库时，需要节点名。

缺省情况是只能从执行了备份的同一节点上的 TSM 中复原一个数据库。有可能在通过 DB2 执行备份（例如，使用 BACKUP DATABASE 命令）期间覆盖 *tsm_nodename*。

## tsm_owner - Tivoli Storage Manager 所有者名称

此参数用于覆盖与 Tivoli Storage Manager（TSM）产品相关的所有者的缺省设置。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 语句边界

**缺省值 [范围]**
Null [任何字符串]

在复原从另一个节点备份至 TSM 的数据库时，需要所有者名称。有可能在通过 DB2 执行备份（例如，使用 BACKUP DATABASE 命令）期间覆盖 *tsm_owner*。

**注**: 所有者名称区分大小写。

缺省情况是只能从执行了备份的同一节点上的 TSM 中复原一个数据库。

## tsm_password - Tivoli Storage Manager 密码

此参数用于重设与 Tivoli Storage Manager（TSM）产品相关的密码的缺省设置。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 语句边界

**缺省值 [范围]**
Null [任何字符串]

在复原从另一个节点备份至 TSM 的数据库时，需要密码。

**注:** 如果在使用 DB2 执行备份（例如，使用 BACKUP DATABASE 命令）期间覆盖了 *tsm_nodename*，那么可能还必须设置 *tsm_password*。

缺省情况是只能从执行了备份的同一节点上的 TSM 中复原一个数据库。有可能在使用 DB2 执行备份期间覆盖 *tsm_nodename*。

## user_exit_status - 用户出口状态指示器

如果设置为 On，那么此参数指示启用数据库管理器用于前滚恢复，并且将使用用户出口程序在数据库管理器调用日志文件时归档和检索日志文件。

**配置类型**
数据库

**参数类型**
参考

## userexit -“启用用户出口”

在版本 9.5 中不推荐使用此参数，但是版本 9.5 之前的数据服务器和客户机仍然使用此参数。DB2 V9.5 数据库管理器将忽略对此配置参数指定的任何值。从 V9.5 开始，**logarchmeth1** 配置参数用于设置 USEREXIT 值。

**注:** 下列信息仅适用于版本 9.5 之前的数据服务器和客户机。

如果启用此参数，那么不管 *logretain* 参数的设置如何，都将执行日志保留日志记录操作。此参数也指示应使用用户出口程序来归档和检索日志文件。

**配置类型**
数据库

**参数类型**
可配置

**缺省值 [范围]**
Off [On; Off]

当日志文件已满时，将归档日志文件。当 ROLLFORWARD 实用程序需要使用这些日志文件来复原数据库时，将检索这些日志文件。

启用 *logretain* 和/或 *userexit* 参数之后，您必须对该数据库进行完全备份。此状态由 *backup_pending* 标志参数指示。

如果取消选择这两个参数，那么前滚恢复对该数据库不可用，因为将不再保留日志。在这种情况下，数据库管理器将删除 *logpath* 目录中的所有日志文件（包括联机归档日志文件）、分配新的活动日志文件，并还原为循环日志记录。

## util_heap_sz - 实用程序堆大小

此参数指示可由备份、复原和装入（包括装入恢复）实用程序同时使用的最大内存量。

**配置类型**
数据库

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
5000 [16 - 524 288 ]

**计量单位**
页 (4 KB)

**分配时间**
数据库管理器实用程序需要时

**释放时间**
当实用程序不再需要内存时

**建议**：使用缺省值，除非实用程序耗尽空间，在这种情况下应增大此值。如果系统上的内存受约束，那么您可能期望降低此参数的值以限制数据库实用程序使用的内存。如果此参数设置得太低，您可能无法并行运行实用程序。应该根据需要动态更新此参数。如果实用程序较少，那么将此参数设置为较小的值。如果实用程序较多，或者实用程序消耗的内存较多，那么应该将此参数设置为较大的值。

## varchar2_compat -“Varchar2 兼容性”数据库配置参数

此参数指示与 VARCHAR2 数据类型相关联的兼容性语义是否应用于所连接的数据库。

**配置类型**
数据库

**参数类型**
参考

值在数据库创建时确定，并且基于用于 VARCHAR2 支持的 DB2_COMPATIBILITY_VECTOR 注册表变量的设置。值不可更改。

## vendoropt - 供应商选项

此参数指定 DB2 在备份、复原或装入副本操作期间可能需要用来与存储系统通信的其他参数。

**配置类型**

数据库

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**缺省值 [范围]**

Null [ ]

**限制** 不能使用 **vendoropt** 配置参数为快照备份或复原操作指定特定于供应商的选项。必须改为使用 backup 或 restore 实用程序的 OPTIONS 参数。

在所配置的支持代理节点的 TSM 环境中，“-fromnode=nodename”选项和“-fromowner=ownername”选项都与“-asnodename=nodename”选项不兼容，不能将它们一起使用。将 -asnodename 选项用于使用代理节点的 TSM 配置，而将另外两个选项用于其他类型的 TSM 配置。有关更多信息，请参阅『配置 Tivoli Storage Manager 客户机』。

## wlm_collect_int - 工作负载管理收集时间间隔配置参数

此参数指定工作负载管理（WLM）统计信息的收集和复位时间间隔（以分钟为单位）。

每隔 x *wlm_collect_int* 分钟（其中 x 是 *wlm_collect_int* 参数的值），就会收集所有工作负载管理统计信息并将它们发送至任何活动统计信息事件监视器，然后复位统计信息。如果存在活动事件监视器，那么将根据该事件监视器的创建方式，将统计信息写入文件或表。如果它不存在，那么将只复位统计信息，而不进行收集。

收集和复位进程是从目录分区启动的。必须在目录分区上指定 *wlm_collect_int* 参数。它不在其他分区上使用。

**配置类型**

数据库

**参数类型**

可联机配置

**缺省值 [范围]**

0 [0（未执行收集），5 - 32 767]

可以使用统计信息事件监视器收集的工作负载管理统计信息来监视短期和长期系统行为。由于可以将结果合并在一起来获得长期行为，所以可以使用较小的时间间隔来同时获得短期系统行为和长期系统行为。但是，由于必须手动合并不同时间间隔中的结果，这将使分析变得复杂。如果不需要手动合并结果，那么较小的时间间隔会导致不必要的开销增大。因此，减小时间间隔以捕获较短期的行为，并且在只分析长期行为就已足够的情况下，增大时间间隔以减少开销。

需要对每个数据库定制时间间隔，而不是对每个 SQL 请求、命令调用或应用程序进行定制。没有其他配置参数需要考虑。

**注**: 所有 WLM 统计信息表函数都返回自上次复位统计信息以来累积的统计信息。将按此配置参数指定的时间间隔定期复位统计信息。

# DB2 管理服务器（DAS）配置参数

## authentication - 认证类型 DAS

确定如何进行以及在何处进行用户认证。

**配置类型**
DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**
可配置

**缺省值 [范围]**
SERVER_ENCRYPT [SERVER_ENCRYPT; KERBEROS_ENCRYPT ]

如果 **authentication** 的值为 SERVER_ENCRYPT，那么会将用户标识和密码从客户机发送至服务器，以便可在服务器上进行认证。将对通过网络发送的用户标识和密码进行加密。

如果值为 KERBEROS_ENCRYPT，那么意味着将在 Kerberos 服务器上使用 Kerberos 认证安全性协议来执行认证。

**注**: 仅在运行 Windows 的服务器上才支持 KERBEROS_ENCRYPT 认证类型。

只能从版本 9 命令行处理器（CLP）更新此参数。

## contact_host - 联系人列表的位置

此参数指定存储由“调度程序”和“运行状况监视器”用于通知的联系人信息的位置。

**配置类型**
DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
Null [任何有效 DB2 管理服务器 TCP/IP 主机名]

该位置定义为 DB2 管理服务器的 TCP/IP 主机名。允许 *contact_host* 位于远程 DAS 上支持在多个 DB2 管理服务器间共享联系人列表。如果未指定 *contact_host*，那么 DAS 假定联系人信息在本地。

此参数只能从版本 8 命令行处理器（CLP）更新。

## das_codepage - DAS 代码页

此参数指示由 DB2 管理服务器使用的代码页。

**配置类型**
DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
Null（任何有效的 DB2 代码页）

如果参数为 null，那么使用系统的缺省代码页。此参数应与本地 DB2 实例的语言环境兼容。否则，DB2 管理服务器不能与 DB2 实例通信。

此参数只能从版本 8 命令行处理器（CLP）更新。

## das_territory - DAS 地域

此参数显示由 DB2 管理服务器使用的地域。

**配置类型**
DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
Null [任何有效 DB2 地域]

如果该参数为 null，那么使用系统的缺省地域。

此参数只能从版本 8 命令行处理器（CLP）更新。

## dasadm_group - DAS 管理权限组名

此参数对 DAS 定义具有 DAS 管理（DASADM）权限的组名。

**配置类型**
DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**
可配置

**缺省值 [范围]**
Null [任何有效组名]

DASADM 权限是 DAS 内的最高级别的权限。

DASADM 权限由在特定操作环境中使用的安全工具确定。

- 对于 Windows 操作系统来说，可将此参数设置为在 Windows 安全性数据库中定义的任何本地组。只要组名的长度不超过 30 个字节，它们就是可接受的组名。如果对此参数指定“NULL”，那么 Administrators 组的所有成员都具有 DASADM 权限。
- 对于 Linux 和 UNIX 系统，如果指定“NULL”作为此参数的值，那么 DASADM 组将缺省为实例所有者的主组。

  如果该值不是“NULL”，那么 DASADM 组可以是任何有效的 UNIX 组名。

此参数只能从版本 8 命令行处理器（CLP）更新。

## db2system - DB2 服务器系统的名称

此参数指定由您的用户和数据库管理员使用以标识 DB2 服务器系统的名称。

**配置类型**

DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**

可联机配置

**缺省值 [范围]**

TCP/IP 主机名（任何有效的系统名称）

应尽可能使此名称在网络中是唯一的。

此名称显示在控制中心的对象树的系统层，以帮助管理员来标识可以从控制中心管理的服务器系统。

当使用“配置助手”的“搜索网络”功能时，DB2 发现返回此名称，并在生成的对象树中的系统层显示该名称。此名称帮助用户识别包含他们希望访问的数据库的系统。在安装时，将 *db2system* 的值设置如下:

- 在 Windows 上，安装程序将它设置成对 Windows 系统指定的计算机名称。
- 在 UNIX 系统上，将它设置成 UNIX 系统的 TCP/IP 主机名。

## diaglevel -“诊断错误捕获级别”配置参数

此参数指定将记录在 `db2dasdiag.log` 文件中的诊断错误的类型。

**配置类型**

数据库管理器

**适用于**

- 带有本地和远程客户机的数据库服务器
- 客户机
- 带有本地客户机的数据库服务器
- 带有本地和远程客户机的分区数据库服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

3 [ 0 - 4 ]

此参数的有效值为:

- 0 - 没有捕获到诊断数据
- 1 - 仅严重错误
- 2 - 所有错误
- 3 - 所有错误和警告
- 4 - 所有错误、警告以及参考消息

**使用说明**

- **diaglevel** 的动态行为不会扩展到所有进程。
- db2sysc DB2 服务器进程可以检测到动态更改，例如，当您对实例连接发出 UPDATE DATABASE MANAGER CONFIGURATION 命令时。
- DB2 客户机和应用程序启动时，将使用 **diaglevel** 配置参数设置且不检测任何动态更改。
- 要帮助解决问题，您可以增大该参数的值以收集更多问题确定数据。

## discover - DAS 发现方式

此参数确定在“DB2 管理服务器”启动时启动的发现方式的类型。

**配置类型**

DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

SEARCH [ DISABLE; KNOWN; SEARCH ]

- 如果 discover = SEARCH，那么管理服务器处理来自客户机的 SEARCH 发现请求。SEARCH 提供由 KNOWN 发现提供的功能的超集。当 discover = SEARCH，管理服务器将处理来自客户机的 SEARCH 和 KNOWN 发现请求。
- 如果 discover = KNOWN，那么管理服务器仅处理来自客户机的 KNOWN 发现请求。
- 如果 discover = DISABLE，那么管理服务器不会处理任何类型的发现请求。此服务器系统的信息本质上已从客户机隐藏。

缺省发现方式为 SEARCH。

此参数只能从版本 8 命令行处理器（CLP）更新。

## exec_exp_task - 执行已到期任务

此参数指定“调度程序”是否将执行已在过去调度但是尚未执行的任务。

**配置类型**

DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**

可配置

**缺省值 [范围]**

No [ Yes; No ]

“调度程序”在启动时，仅检测到期任务。例如，如果有一个作业调度为在每个星期六运行，并且“调度程序”在星期五关闭并在星期一重新启动，那么对星期六调度的作业现在是在过去调度的作业。如果将 *exec_exp_task* 设置为 Yes，那么当“调度程序”重新启动时，将运行星期六作业。

此参数只能从版本 8 命令行处理器（CLP）更新。

## jdk_path - Java 软件开发者工具箱安装路径 DAS

此参数指定要用于运行 DB2 管理服务器函数的 Java 软件开发者工具箱（SDK）的安装目录。

**配置类型**

DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**

可联机配置

**传播类** 立即

**缺省值 [范围]**

缺省 Java 安装路径（任何有效路径）

Java 解释器使用的是根据此参数的环境变量计算出来的。

在 Windows 操作系统上，在 DB2 安装期间，Java 文件（如果需要）放置在 sqllib 目录（在 java\jdk 中）下面: 然后将 *jdk_path* 配置参数设置为 sqllib\java\jdk。DB2 永远不会真正在 Windows 平台上安装 Java，只是将文件放置在 sqllib 目录下面，并且不管是否已安装 Java 都会这样做。

此参数只能从版本 8 命令行处理器（CLP）更新。

## sched_enable - 调度程序方式

此参数指示“调度程序”是否由管理服务器启动。

**配置类型**

DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**

可配置

**缺省值 [范围]**

Off [On; Off ]

“调度程序”允许工具（例如，任务中心）来在管理服务器上调度和执行任务。

此参数只能从版本 8 命令行处理器（CLP）更新。

## sched_userid - 调度程序用户标识

此参数指定由“调度程序”使用以与工具目录数据库连接的用户标识。仅当工具目录数据库对于 DB2 管理服务器是远程时，此参数才是切题的。

**配置类型**
DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**
*参考*

**缺省值 [范围]**
Null [任何有效用户标识]

由“调度程序”使用以与远程工具目录数据库连接的用户标识和密码是使用 db2admin 命令指定的。

## smtp_server - SMTP 服务器

当“调度程序”打开时，此参数标识“调度程序”将用来发送电子邮件和寻呼机通知的 SMTP 服务器。

**配置类型**
DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**
可联机配置

**传播类** 立即

**缺省值 [范围]**
Null（任何有效的 SMTP 服务器 TCP/IP 主机名）

此参数由“调度程序”和“运行状况监视器”使用。

此参数只能从版本 8 命令行处理器（CLP）更新。

## toolscat_db - 工具目录数据库

此参数指示由“调度程序”使用的工具目录数据库。

**配置类型**
DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**
可配置

**缺省值 [范围]**
Null [任何有效数据库别名]

此数据库必须在由 *toolscat_inst* 指定的实例的数据库目录中。

此参数只能从版本 8 命令行处理器（CLP）更新。

## toolscat_inst - 工具目录数据库实例

此参数指示“调度程序”以及 toolscat_db 和 toolscat_schema 使用的用于标识工具目录数据库的实例名。

**配置类型**

DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**

可配置

**缺省值 [范围]**

Null [任何有效实例]

工具目录数据库包含任务中心和控制中心创建的任务信息。工具目录数据库必须列示在此配置参数指定的实例的数据库目录中。数据库可以是本地数据库或远程数据库。如果工具目录数据库是本地的，那么必须为 TCP/IP 配置实例。如果数据库是远程的，那么数据库目录中编目的数据库分区必须是 TCP/IP 节点。

此参数只能从版本 8 命令行处理器（CLP）更新。

## toolscat_schema - 工具目录数据库模式

此参数指示由“调度程序”使用的工具目录数据库的模式。

**配置类型**

DB2 管理服务器

**适用于** DB2 管理服务器

**参数类型**

可配置

**缺省值 [范围]**

Null [任何有效模式]

此模式用来唯一标识数据库内的一组工具目录表和视图。

此参数只能从版本 8 命令行处理器（CLP）更新。

# 第 5 部分 附录

© Copyright IBM Corp. 1993, 2010

# 附录 A. DB2 技术信息概述

可以通过下列工具和方法获取 DB2 技术信息:

- *DB2 信息中心*
  - 主题（任务、概念和参考主题）
  - DB2 工具的帮助
  - 样本程序
  - 教程
- DB2 书籍
  - PDF 文件（可下载）
  - PDF 文件（在 DB2 PDF DVD 中）
  - 印刷版书籍
- 命令行帮助
  - 命令帮助
  - 消息帮助

**注**: *DB2 信息中心*主题的更新频率比 PDF 书籍或硬拷贝书籍的更新频率高。要获取最新信息，请安装可用的文档更新，或者参阅 ibm.com® 上的 *DB2 信息中心*。

可以在线访问 ibm.com 上的其他 DB2 技术信息，如技术说明、白皮书和 IBM Redbooks® 出版物。访问位于以下网址的 DB2 信息管理软件库站点: http://www.ibm.com/software/data/sw-library/。

## 文档反馈

我们非常重视您对 DB2 文档的反馈。如果您想就如何改善 DB2 文档提出建议，请发送电子邮件至 db2docs@ca.ibm.com。DB2 文档小组会阅读您的所有反馈，但不能直接答复您。请尽可能提供具体的示例，这样我们才能更好地了解您所关心的问题。如果您要提供有关具体主题或帮助文件的反馈，请加上标题和 URL。

请不要用以上电子邮件地址与 DB2 客户支持机构联系。如果您遇到文档不能解决的 DB2 技术问题，请与您当地的 IBM 服务中心联系以获得帮助。

如果愿意帮助 IBM 使 IBM 信息管理产品更易于使用，请参与可消费性调查: http://www.ibm.com/software/data/info/consumability-survey/。

# 硬拷贝或 PDF 格式的 DB2 技术库

下列各表描述 IBM 出版物中心（网址为 www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss）所提供的 DB2 资料库。可从 www.ibm.com/support/docview.wss?uid=swg27009727 和 www.ibm.com/support/docview.wss?uid=swg27009728 分别下载 PDF 格式的 DB2 V9.5 手册的英文版本和翻译版本。

尽管这些表标识书籍有印刷版，但可能未在您所在国家或地区提供。

© Copyright IBM Corp. 1993, 2010

每次更新手册时，表单号都会递增。确保您正在阅读下表中列示的手册的最新版本。

**注:** *DB2 信息中心*的更新频率比 PDF 或硬拷贝书籍的更新频率高。

*表 72. DB2 技术信息*

| 书名 | 书号 | 是否提供印刷版 | 最近一次更新时间 |
|---|---|---|---|
| *Administrative API Reference* | SC23-5842-03 | 是 | 2010 年 12 月 |
| *Administrative Routines and Views* | SC23-5843-03 | 否 | 2010 年 12 月 |
| *Call Level Interface Guide and Reference, Volume 1* | SC23-5844-03 | 是 | 2010 年 12 月 |
| *Call Level Interface Guide and Reference, Volume 2* | SC23-5845-03 | 是 | 2010 年 12 月 |
| *Command Reference* | SC23-5846-03 | 是 | 2010 年 12 月 |
| *《数据移动指南和参考》* | S151-0617-02 | 是 | 2010 年 12 月 |
| *《数据恢复及高可用性指南与参考》* | S151-0619-02 | 是 | 2010 年 12 月 |
| *《数据服务器、数据库和数据库对象指南》* | S151-0612-02 | 是 | 2010 年 12 月 |
| *《数据库安全性指南》* | S151-0614-02 | 是 | 2010 年 12 月 |
| *Developing ADO.NET and OLE DB Applications* | SC23-5851-02 | 是 | 2009 年 4 月 |
| *Developing Embedded SQL Applications* | SC23-5852-02 | 是 | 2009 年 4 月 |
| *Developing Java Applications* | SC23-5853-03 | 是 | 2010 年 12 月 |
| *Developing Perl and PHP Applications* | SC23-5854-02 | 否 | 2009 年 4 月 |
| *Developing User-defined Routines (SQL and External)* | SC23-5855-03 | 是 | 2010 年 12 月 |
| *Getting Started with Database Application Development* | GC23-5856-03 | 是 | 2010 年 12 月 |
| *《Linux 和 Windows 上的 DB2 安装和管理入门》* | G151-0623-02 | 是 | 2010 年 12 月 |
| *《国际化指南》* | S151-0616-01 | 是 | 2009 年 4 月 |
| *《消息参考，第 1 卷》* | G151-0632-00 | 否 | |
| *《消息参考，第 2 卷》* | G151-0633-00 | 否 | |
| *《迁移指南》* | G151-0622-02 | 是 | 2010 年 12 月 |
| *《Net Search Extender 管理和用户指南》* | S151-0760-01 | 是 | 2009 年 4 月 |
| *《分区和集群指南》* | S151-0615-02 | 是 | 2010 年 12 月 |

表 72. DB2 技术信息 (续)

| 书名 | 书号 | 是否提供印刷版 | 最近一次更新时间 |
| --- | --- | --- | --- |
| *Query Patroller Administration and User's Guide* | SC23-8507-01 | 是 | 2009 年 4 月 |
| 《IBM 数据服务器客户机快速入门》 | G151-0625-02 | 否 | 2010 年 12 月 |
| 《DB2 服务器快速入门》 | G151-0624-02 | 是 | 2010 年 12 月 |
| *Spatial Extender and Geodetic Data Management Feature User's Guide and Reference* | SC23-8508-02 | 是 | 2009 年 4 月 |
| *SQL Reference, Volume 1* | SC23-5861-03 | 是 | 2010 年 12 月 |
| *SQL Reference, Volume 2* | SC23-5862-03 | 是 | 2010 年 12 月 |
| 《系统监视器指南和参考》 | S151-0618-02 | 是 | 2010 年 12 月 |
| *Text Search Guide* | SC23-5866-02 | 是 | 2010 年 12 月 |
| 《故障诊断指南》 | G151-0621-02 | 否 | 2010 年 12 月 |
| 《调整数据库性能》 | S151-0613-02 | 是 | 2010 年 12 月 |
| 《Visual Explain 教程》 | S151-0634-00 | 否 | |
| 《新增内容》 | S151-0629-02 | 是 | 2010 年 12 月 |
| *Workload Manager Guide and Reference* | SC23-5870-03 | 是 | 2010 年 12 月 |
| 《pureXML 指南》 | S151-0630-02 | 是 | 2010 年 12 月 |
| 《XQuery 参考》 | S151-0631-01 | 否 | 2009 年 4 月 |

表 73. 特定于 DB2 Connect 的技术信息

| 书名 | 书号 | 是否提供印刷版 | 最近一次更新时间 |
| --- | --- | --- | --- |
| 《DB2 Connect 个人版快速入门》 | G151-0627-02 | 是 | 2010 年 12 月 |
| 《DB2 Connect 服务器快速入门》 | G151-0628-02 | 是 | 2010 年 12 月 |
| 《DB2 Connect 用户指南》 | S151-0626-02 | 是 | 2010 年 12 月 |

表 74. Information Integration 技术信息

| 书名 | 书号 | 是否提供印刷版 | 最近一次更新时间 |
| --- | --- | --- | --- |
| *Information Integration: Administration Guide for Federated Systems* | SC19-1020-01 | 是 | |
| *Information Integration: ASNCLP Program Reference for Replication and Event Publishing* | SC19-1018-02 | 是 | |

表 74. Information Integration 技术信息 (续)

| 书名 | 书号 | 是否提供印刷版 | 最近一次更新时间 |
| --- | --- | --- | --- |
| *Information Integration: 《联合数据源配置指南》* | S151-0468-00 | 否 | |
| *Information Integration: 《SQL 复制指南和参考》* | S151-0475-00 | 是 | |
| *Information Integration: Introduction to Replication and Event Publishing* | GC19-1028-01 | 是 | |

## 订购印刷版的 DB2 书籍

如果您需要印刷版的 DB2 书籍，可以在许多（但不是所有）国家或地区在线购买。无论何时都可以从当地的 IBM 代表处订购印刷版的 DB2 书籍。请注意，*DB2 PDF 文档* DVD 上的某些软拷贝书籍没有印刷版。例如，*DB2 消息参考*的任何一卷都没有提供印刷版书籍。

只要支付一定费用，就可以从 IBM 获取 DB2 PDF 文档 DVD，该 DVD 包含许多印刷版 DB2 书籍的内容。根据您下订单的位置，您可能能够从 IBM 出版物中心在线订购书籍。如果在线订购在您所在国家或地区不可用，您始终可以从当地的 IBM 代表处订购印刷版 DB2 书籍。注意，并非 DB2 PDF 文档 DVD 上的所有书籍都有印刷版。

**注**：最新最完整的 DB2 文档保留在 DB2 信息中心中，网址如下：http://publib.boulder.ibm.com/infocenter/db2luw/v9r5。

要订购印刷版的 DB2 书籍：

- 要了解您是否可从所在国家或地区在线订购印刷版的 DB2 书籍，可查看 IBM 出版物中心站点，网址为：http://www.ibm.com/shop/publications/order。必须先选择国家、地区或语言才能访问出版物订购信息，然后再按照针对您所在位置的订购指示信息进行订购。
- 要从当地的 IBM 代表处订购印刷版的 DB2 书籍：
  1. 从下列其中一个 Web 站点找到当地代表处的联系信息：
     - IBM 全球联系人目录，网址为 www.ibm.com/planetwide。
     - IBM 出版物 Web 站点，网址为 http://www.ibm.com/shop/publications/order。必须先选择国家、地区或语言才能访问对应您的所在地的出版物主页。在此页面中访问“关于此站点”链接。
  2. 请在致电时说明您想订购 DB2 出版物。
  3. 请向您当地的代表提供想要订购的书籍的书名和书号。有关书名和书号的信息，请参阅第 565 页的『硬拷贝或 PDF 格式的 DB2 技术库』。

## 从命令行处理器显示 SQL 状态帮助

DB2 返回描述 SQL 语句执行结果的 SQLSTATE。SQLSTATE 帮助说明 SQL 状态和 SQL 状态类代码的含义。

要调用 SQL 状态帮助，请打开命令行处理器并输入:

```
? sqlstate or ? class code
```

其中，*sqlstate* 表示有效的 5 位 SQL 状态，*class code* 表示该 SQL 状态的前 2 位。例如，`? 08003` 显示 08003 SQL 状态的帮助，而 `? 08` 显示 08 类代码的帮助。

## 访问不同版本的 DB2 信息中心

对于 DB2 版本 9.8 主题，*DB2 信息中心* URL 为 http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/。

对于 DB2 版本 9.7 主题，*DB2 信息中心* URL 为: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/。

对于 DB2 V9.5 主题，*DB2 信息中心* URL 为 http://publib.boulder.ibm.com/infocenter/db2luw/v9r5。

对于 DB2 V9.1 主题，*DB2 信息中心* URL 为 http://publib.boulder.ibm.com/infocenter/db2luw/v9/。

对于 DB2 V8 主题，请转至 *DB2 信息中心* URL: http://publib.boulder.ibm.com/infocenter/db2luw/v8/。

## 在 DB2 信息中心中以您的首选语言显示主题:

DB2 信息中心尝试以您在浏览器首选项中指定的语言显示主题。如果未提供主题的首选语言翻译版本，那么 DB2 信息中心将显示该主题的英文版。

- 要在 Internet Explorer 浏览器中以您的首选语言显示主题:
  1. 在 Internet Explorer 中，单击**工具** —> **Internet 选项** —> **语言...**按钮。“语言首选项”窗口打开。
  2. 确保您的首选语言被指定为语言列表中的第一个条目。
     - 要将新语言添加至列表，请单击**添加...** 按钮。

       **注**: 添加语言并不能保证计算机具有以首选语言显示主题所需的字体。
     - 要将语言移至列表顶部，请选择该语言并单击**上移**按钮直到该语言成为语言列表中的第一个条目。
  3. 清除浏览器高速缓存然后刷新页面以便以首选语言显示 DB2 信息中心。
- 要在 Firefox 或 Mozilla 浏览器中以首选语言显示主题:
  1. 在**工具** —> **选项** —> **高级**对话框中的**语言**部分中选择按钮。“语言”面板将显示在“首选项”窗口中。
  2. 确保您的首选语言被指定为语言列表中的第一个条目。

– 要将新语言添加至列表，请单击**添加...** 按钮以从“添加语言”窗口中选择一种语言。
– 要将语言移至列表顶部，请选择该语言并单击**上移**按钮直到该语言成为语言列表中的第一个条目。

3. 清除浏览器高速缓存然后刷新页面以便以首选语言显示 DB2 信息中心。

在某些浏览器和操作系统组合上，可能还必须将操作系统的区域设置更改为您选择的语言环境和语言。

## 更新安装在您的计算机或内部网服务器上的 DB2 信息中心

如果已经在本地安装了 DB2 信息中心，那么您可以从 IBM 获取文档更新并安装。

要更新在本地安装的 *DB2 信息中心*，您需要:

1. 停止计算机上的 *DB2 信息中心*，然后以独立方式重新启动信息中心。如果以独立方式运行信息中心，那么网络上的其他用户将无法访问信息中心，因而您可以应用更新。非管理员和非 Root 用户 *DB2 信息中心*始终以独立方式运行。。
2. 使用“更新”功能部件来查看可用的更新。如果有您希望安装的更新，那么请使用“更新”功能部件来获取并安装这些更新。

   **注:** 如果您所在的环境要求在一台未连接至因特网的机器上安装 *DB2 信息中心*更新，那么必须使用一台已连接至因特网的机器将更新站点镜像至本地文件系统，并安装 *DB2 信息中心*。如果网络中有许多用户将安装文档更新，那么可以通过在本地也为更新站点制作镜像并为更新站点创建代理来缩短每个人执行更新所需要的时间。
   如果提供了更新包，请使用“更新”功能部件来获取这些更新包。但是，只有在独立方式下才能使用更新功能。
3. 停止独立信息中心，然后在计算机上重新启动 *DB2 信息中心*。

**注:** 在 Windows Vista 上，必须以管理员身份才能运行下面所列示的命令。要启动具有全面管理员特权的命令提示符或图形工具，请右键单击快捷方式，然后选择**以管理员身份运行**。

要更新安装在您的计算机或内部网服务器上的 *DB2 信息中心*:

1. 停止 *DB2 信息中心*。
   - 在 Windows 上，单击**开始 → 控制面板 → 管理工具 → 服务**。右键单击 **DB2 信息中心**服务，并选择**停止**。
   - 在 Linux 上，输入以下命令:

     ```
     /etc/init.d/db2icdv95 stop
     ```
2. 以独立方式启动信息中心。
   - 在 Windows 上:
     a. 打开命令窗口。
     b. 浏览至信息中心的安装位置。缺省情况下，*DB2 信息中心*安装在 *Program_files*\IBM\DB2 Information Center\Version 9.5 目录中，其中 *Program_files* 表示 Program Files 目录的位置。
     c. 从安装目录浏览至 doc\bin 目录。

d. 运行 help_start.bat 文件:

```
help_start.bat
```

- 在 Linux 上:
  a. 浏览至信息中心的安装位置。缺省情况下，*DB2 信息中心*安装在 /opt/ibm/db2ic/V9.5 目录中。
  b. 从安装目录浏览至 doc/bin 目录。
  c. 运行 help_start 脚本:

```
help_start
```

系统缺省 Web 浏览器将启动以显示独立信息中心。

3. 单击**更新**按钮（ ）。在信息中心的右边面板上，单击**查找更新**。 将显示现有文档的更新列表。
4. 要启动安装进程，请检查您要安装的选项，然后单击**安装更新**。
5. 在安装进程完成后，请单击**完成**。
6. 要停止独立信息中心，请执行下列操作:
   - 在 Windows 上，浏览至安装目录的 doc\bin 目录并运行 help_end.bat 文件:

     ```
     help_end.bat
     ```

     **注**: help_end 批处理文件包含安全地终止使用 help_start 批处理文件启动的进程所需的命令。不要使用 Ctrl-C 或任何其他方法来终止 help_start.bat。
   - 在 Linux 上，浏览至安装目录的 doc/bin 目录并运行 help_end 脚本:

     ```
     help_end
     ```

     **注**: help_end 脚本包含安全地终止使用 help_start 脚本启动的进程所需的命令。不要使用任何其他方法来终止 help_start 脚本。
7. 重新启动 *DB2 信息中心*。
   - 在 Windows 上，单击**开始 → 控制面板 → 管理工具 → 服务**。右键单击 **DB2 信息中心**服务，并选择**启动**。
   - 在 Linux 上，输入以下命令:

     ```
     /etc/init.d/db2icdv95 start
     ```

更新后的 *DB2 信息中心*将显示新的以及更新后的主题。

# DB2 教程

DB2 教程帮助您了解 DB2 产品的各个方面。这些课程提供了逐步指示信息。

## 开始之前

可从信息中心查看 XHTML 版的教程: http://publib.boulder.ibm.com/infocenter/db2help/ 。

某些课程使用了样本数据或代码。有关其特定任务的任何先决条件的描述，请参阅教程。

## DB2 教程

要查看教程，请单击标题。

*《pureXML 指南》*中的『**pureXML**』

设置 DB2 数据库以存储 XML 数据以及对本机 XML 数据存储执行基本操作。

*《Visual Explain 教程》*中的『**Visual Explain**』

使用 Visual Explain 来分析、优化和调整 SQL 语句以获取更好的性能。

## DB2 故障诊断信息

提供了很多故障诊断和问题确定信息以帮助您使用 DB2 数据库产品。

**DB2 文档**

故障诊断信息可在 DB2 信息中心的“DB2 故障诊断指南”或“数据库基础”部分中找到。可在该处找到有关如何使用 DB2 诊断工具和实用程序来隔离和找出问题的信息、某些最常见问题的解决方案以及有关如何解决使用 DB2 数据库产品时可能遇到的问题的建议。

**DB2 技术支持 Web 站点**

如果您遇到了问题并且想要获取查找可能的原因和解决方案的帮助，请参阅 DB2 技术支持 Web 站点。该“技术支持”站点具有指向最新 DB2 出版物、技术说明、授权程序分析报告（APAR 或错误修订）、修订包和其他资源的链接。可搜索此知识库并查找问题的可能解决方案。

请访问 DB2 技术支持 Web 站点: http://www.ibm.com/software/data/db2/support/db2_9/。

## 条款和条件

如果符合以下条款和条件，那么授予您使用这些出版物的准用权。

**个人使用**: 只要保留所有的专有权声明，您就可以为个人、非商业使用复制这些出版物。未经 IBM 明确同意，您不可以分发、展示或制作这些出版物或其中任何部分的演绎作品。

**商业使用**: 只要保留所有的专有权声明，您就可以仅在企业内复制、分发和展示这些出版物。未经 IBM 明确同意，您不可以制作这些出版物的演绎作品，或者在您的企业外部复制、分发或展示这些出版物或其中的任何部分。

除非本准用权中有明确授权，不得把其他准用权、许可或权利（无论是明示的还是暗含的）授予其中包含的出版物或任何信息、数据、软件或其他知识产权。

当使用这些出版物损害了 IBM 的利益，或者根据 IBM 的规定，未正确遵守上述指导说明时，那么 IBM 保留自主决定撤销本文授予的准用权的权利。

您不可以下载、出口或再出口本信息，除非完全遵守所有适用的法律和法规，包括所有美国出口法律和法规。

IBM 对这些出版物的内容不作任何保证。这些出版物“按现状”提供，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的关于适销和适用于某种特定用途的保证。

# 附录 B. 声明

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，那么由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation
Licensing
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

**本条款不适用英国或任何这样的条款与当地法律不一致的国家或地区**: International Business Machines Corporation“按现状”提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本资料中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本文档可能提供非 IBM Web 站点和资源的链接或引用。IBM 对于任何非 IBM Web 站点或第三方资源不作任何声明、保证或其他承诺，即使本文档可能引用了这些 Web 站点或第三方资源，或者可从本文档访问或链接到这些 Web 站点或第三方资源。到某个非 IBM Web 站点的链接并不意味着 IBM 认可此类 Web 站点的内容或使用或其所有者。此外，IBM 不是您与任何第三方签署协议的任何交易的一方，也不对任何交易负责，即使您从某个 IBM 站点了解到此类第三方或使用到此类第三方的链接时亦如此。因此，您需要承认并同意，IBM 不对此类外部站点或资源的可用性负责，也不对可从那些站点或资源上获得的任何内容、服务、产品或其他资料承担任何责任或义务。第三方提供的任何软件须遵守该软件随附的许可证的条款和条件。

© Copyright IBM Corp. 1993, 2010

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: （i）允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Canada Limited
U59/3600
3600 Steeles Avenue East
Markham, Ontario L3R 9Z7
CANADA

只要遵守适当的条款和条件，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息可能包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名称都是虚构的，与实际商业企业所用的名称和地址的任何雷同纯属巧合。

版权许可:

本信息可能包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。

凡这些样本程序的每份拷贝或其任何部分或任何衍生产品，都必须包括如下版权声明:

©（*贵公司的名称*）（*年份*）. 此部分代码是根据 IBM 公司的样本程序衍生出来的。© Copyright IBM Corp. （*输入年份*）. All rights reserved.

## 商标

IBM、IBM 徽标和 ibm.com 是 International Business Machines Corp. 在全世界许多管辖区域的商标或注册商标。其他产品和服务名称可能是 IBM 或其他公司的商标。在版权和商标信息的 Web 页面上提供了 IBM 商标的当前列表，网址为: www.ibm.com/legal/copytrade.shtml。

下列术语是其他公司的商标或注册商标。

- Linux 是 Linus Torvalds 在美国和/或其他国家或地区的注册商标。
- Java 和所有基于 Java 的商标是 Sun Microsystems, Inc. 在美国和/或其他国家或地区的商标。
- UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。
- Intel、Intel 徽标、Intel Inside、Intel Inside 徽标、Intel Centrino、Intel Centrino 徽标、Celeron、Intel Xeon、Intel SpeedStep、Itanium 和 Pentium 是 Intel Corporation 或其子公司在美国和其他国家或地区的商标或注册商标。
- Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

其他公司、产品或服务名称可能是其他公司的商标或服务标记。

# 索引

## [ A ]


安全标号（LBAC）
  策略
    名称长度 323
  名称长度 323
  组件名称程度 323
安全性
  插件
    配置参数 431, 433, 471, 473


## [ B ]


帮助
  配置语言 569
  SQL 语句 569
绑定
  更改配置参数 407, 408
  数据库实用程序 118
备份
  跟踪已修改的页 550
本地数据库目录
  查看 127
  描述 105
标识
  长度限制 323
标识列
  示例 211
  修改 235
  与序列比较 295, 297
  在新表上定义 211
表
  标识列 211
  别名 237
  不匹配 231
  查看定义 237
  常规 205
  重命名表 236
  创建
    概述 230
  从属 245
  定义
    引用约束 251
  多维集群 205
  范围集群 205
  方案 238
  分区 205
  父代 245
  共享文件句柄 34
  估计大小要求 108
  行压缩 44, 218
表 (续)
  后代 245
  基本 205
  检查约束
    概述 213
    类型 245
  结果 205
  空间需求 214
  类型 205
  临时 205
  描述 205
  目标 231
  全局临时 205
  缺省列 213
  删除 237
  删除列 234
  设计 207
  设计概念 207
  生成的列 211
  示例 238
  数据类型定义 213
  刷新 234
  添加列 234
  唯一约束 213
  修改 233
  修改 DEFAULT 子句列定义 234
  压缩
    列值 222
    NULL 222
  页大小 215
  引用完整性 213
  映射至表空间 154
  用户 216
  源 231
  摘要 205
  主键 213
  追加方式 205
  自引用 245
  Unicode 表和数据注意事项 214
表达式
  NEXT VALUE 293
  PREVIOUS VALUE 293
表分区
  数据组织方案 230
表检查约束 245
表空间
  不使用文件系统高速缓存 160, 163
  重命名 187
  初始 168
  创建 172
  磁盘 I/O 注意事项 167


© Copyright IBM Corp. 1993, 2010

表空间 *(续)*
调整容器的大小 183
改变 175
自动存储器 187
DMS 容器 176
SMS 容器 176
工作负载注意事项 153
类型 139
SMS 或 DMS 152
临时
建议 150
描述 137
容器
扩展 183
文件示例 172
删除 189
设备容器示例 172
设计 138
数据库管理的空间（DMS） 142
添加
容器 176
系统管理的空间（SMS） 140
系统临时 172
性能 188
选择扩展数据块大小 166
页大小 167
映射 144
映射至表 154
用户临时 172
转换状态 188
自动存储器 34, 148
自动调整大小 36, 155
表空间映射 144
表压缩
概述 217
缺省值 217
别名
创建 119
概述 237
链接过程 237
删除 127
并发事务 120
并行控制
最大活动应用程序数 525
并行性
配置参数
dft_degree 498
intra_parallel 453
max_querydegree 458
I/O
独立磁盘冗余阵列（RAID）设备 188

## [ C ]

参考约束 243
描述 245, 249
参考约束 *(续)*
设计 256
插入规则
引用约束 245
用于引用完整性 245
查询
语句堆大小配置参数 549
查询工作负载
表空间设计 153
查询优化
配置参数 423
常规表
与其他表类型比较 205
长字段
高速缓存行为 153
程序包
不可用 256
池中初始代理程序数配置参数 463
重新路由客户机
LDAP 95
重新平衡
容器 176
初始受防护进程数配置参数 463
处理器
添加 3
触发操作
编码 283
受支持的 SQL PL 语句 284
条件
WHEN 子句 283
触发器
编写触发操作 283
触发事件 280
创建 287
访问旧列值和新列值 285
后
示例 277
激活时间 281
级联 275
交互 253, 288
类型 276
粒度规则 280
描述 275
前
示例 276
删除 288
设计 278
示例
定义操作 290
定义业务规则 291
防止对表进行操作 292
条件 283
修改 288
引用旧表结果集和新表结果集 286
与检查约束比较 251
约束, 交互 253, 288

触发器 (续)
最大名称长度 323
AFTER 子句 281
BEFORE 子句 281
INSTEAD OF
示例 277
INSTEAD OF 子句 281
创建
LDAP 用户 93
从属表
概述 245
从属行
概述 245
存储路径
监视 115
自动
添加 42, 114
存储器
压缩
表 217
行 44, 218
自动
表空间 34, 148
自动, 对于数据库 39, 112

## [ D ]

大对象（LOB）
高速缓存行为 153
大内存页支持
AIX 4
大小限制
标识长度 323
SQL 323
大小要求
估计 108
带类型视图
描述 305
修改 312
代理程序
代理程序优先级配置参数 428
概述 33
配置 32
配置参数影响多个 422
最大并发代理程序数 460
最大代理程序数 459
applheapsz 配置参数 484
aslheapsz 配置参数 429
代理程序池大小
控制 464
代理程序池大小配置参数 464
代码页
数据库配置参数 491
登台表
创建 232
删除 238
调用级接口（CLI）
绑定至数据库 118
调整分区
确定 28
订购 DB2 书籍 568
定界标识
命名规则 319
独立磁盘冗余阵列（RAID）
优化性能 188
堆
配置 30
对象名
规则 319
对组的落实次数配置参数 529
多个实例 60
概述 12
Windows 12, 61
多个 DB2 副本
概述 7
缺省 IBM 数据库客户机接口副本 7
设置缺省实例 11
同时运行多个实例 14, 66
多维集群（MDC）表
延迟索引清除 55
与其他表类型比较 205

## [ F ]

发现方式配置参数 443
发现服务器实例配置参数 443
发现功能
发现方式配置参数 443
范围集群表
与其他表类型比较 205
方案
基于时间的更新检测 241
非缓冲区 I/O
启用/禁用 160
非集群索引 265
非唯一索引 265
分布式关系数据库
连接至 120
远程工作单元 120
分割集 144
DMS 表空间 176
分区表
创建 231
与其他表类型比较 205
分区数据库环境
自调整内存 27, 28
父表
概述 245
父行
概述 245
父键
概述 245

复位压缩字典 42, 219
复原数据库
含义 115
复制模式
操作，重新启动 199

## [ G ]

改变具体化查询表属性 233
概要文件
注册表 333
高速缓存
表空间的文件系统 160
更大的 RID
调整系统临时表空间页大小 151
更改数据库日志路径配置参数 531
更新
DB2 信息中心 570
更新规则
引用约束 245
用于引用完整性 245
共享文件句柄表 34
供应商代码
在受防护供应商进程中 34
工作单元（UOW）
应用程序导向的分布式 123
语义 126
故障诊断
教程 572
联机信息 572

## [ H ]

行
从属 245
父代 245
更改标记 225
后代 245
自引用 245
行标识（RID_BIT 或 RID） 223
行标识（RID_BIT 或 RID）内置函数 223
行更改时间戳记 227
行压缩
更新日志 221
详细信息 44, 217, 218
字典
描述 219
自动 17
后触发器 277
环境变量
概述 341
概要文件注册表 333
设置
Linux 338
UNIX 338
环境变量 *(续)*
设置 *(续)*
Windows 337
声明 335
Linux 338
UNIX
设置 338
Windows 337
缓冲池
创建 134
对查询优化的影响 423
关于 131
内存（保护） 133
删除 136
设计 131
修改 135
恢复
备份暂挂指示器配置参数 488
复原暂挂配置参数 542
前滚暂挂指示器配置参数 542
日志保留状态指示器配置参数 516
视图
不可用 312
数据库的备份次数配置参数 532
索引重新创建时间配置参数 449, 510
用户出口状态指示器配置参数 552
摘要表
不可用 236
自动重新启动启用配置参数 487
“装入恢复会话的缺省数目”配置参数 499
恢复范围和软检查点时间间隔配置参数 546
恢复历史记录文件
保留期配置参数 541
恢复日志
在创建数据库期间分配 108

## [ J ]

基本表
与其他表类型比较 205
基于标号的访问控制（LBAC）
限制 323
基于时间的更新检测 226
方案 241
集群管理器
集群管理器名称配置参数 434
集群索引 265
准则和注意事项 266
检查选项
在视图中
示例 306
检查约束 243
描述 213
设计 250
与前触发器比较 251

键
父代约束 245
外键约束 245
教程
故障诊断 572
问题确定 572
Visual Explain 571
节点
连接耗用时间 435
协调代理程序 457
最大时差 458
节点级概要文件注册表 333
节点间的最大时差配置参数 458
节点连接重试次数配置参数 456
节点目录
查看 105
描述 105
为数据库分区编目 105
节点配置文件
创建 106
结果表
与其他表类型比较 205
进程技术模型
概述 33
简化配置 32
警报摘要
DB2 运行状况监视器 115
镜像日志路径配置参数 530
具体化查询表 (MQT)
改变属性 233
删除 238
刷新数据 234

## [ K ]

可插入视图
使用 310
可更新视图
使用 310
可删除视图
描述 309
客户机接口副本
缺省值 7
客户机支持
客户机 I/O 块大小配置参数 467
TCP/IP 服务名称配置参数 474
客户机 I/O 块大小配置参数 467
空白数据类型 213
库函数
在受防护方式进程中运行 34
跨越的日志数配置参数 536
块结构化设备 172
扩展数据块大小
表空间 166

## [ L ]

乐观锁定
方案 A
启用乐观锁定 238
方案 C
使用隐式隐藏的列 240
概述 223
关于 223
行更改标记 230
基于时间的更新检测 226, 230
计划启用 229
启用 230
使用方案 238
使用 RID() 函数 230
条件 229
限制 224
隐式隐藏的列 224, 230
LBAC 注意事项 224
类型表
与其他表类型比较 205
联合配置参数 446
联合数据库
系统支持配置参数 446
联机事务处理（OLTP）
表空间设计 153
联机维护 19
连接
耗用时间 435
连接耗用时间配置参数 435
连接状态
描述 125
应用程序进程 124
列
订购 221
定义
修改 235
改变 235
隐式隐藏的 224, 230
列数据
约束 212
列数据类型
指定 207
列属性
更改 234
临时表
全局（用户定义的） 231
与其他表类型比较 205
临时表空间
建议 150
落实
对组的落实次数 (mincommit) 529

## [ M ]

每个事务的最大日志配置参数 525
每个应用程序打开的最大数据库文件数配置参数 526
命令行处理器（CLP）
  绑定至数据库 118
命名规则
  本地语言 320
  定界标识和对象名 319
  模式名限制 195
  限制 317
  用户、用户标识和用户组 320
  DB2 对象 318
  Unicode 321
模式
  重新启动失败的复制模式操作 199
  创建 195
  复制 196
  故障诊断提示 196
  描述 191, 194
  命名限制和建议 195
  删除 201
  设计 192
  db2move COPY 错误 199
目录
  本地数据库
    查看 127
    描述 105
  节点
    查看 105
    为数据库分区编目 105
  实例 59
  系统数据库
    查看 127
    描述 105
目录高速缓存大小配置参数 490
目录高速缓存支持配置参数
  描述 442
目录模式
  扩展
    IBM Tivoli Directory Server 83
    Sun One Directory Server 86
目录视图
  描述 306

## [ N ]

内存
  程序包高速缓存大小配置参数 539
  分配时 20
  内存参数之间的交互 23
  排序堆大小配置参数 547
  排序堆阈值配置参数 468
  配置 30
    自调整内存 20
  实例内存配置参数 451
内存 (续)
  应用程序内存配置参数 483
  语句堆大小配置参数 549
  组织使用 20
  applheapsz 配置参数 484
  aslheapsz 配置参数 429
  dbheap 配置参数 496
内存调整器
  分区数据库环境 28
内存配置
  概述 33
内置函数 223

## [ P ]

排序
  共享排序的排序堆阈值 545
  排序堆大小配置参数 547
  排序堆阈值配置参数 468
派生表
  概述 245
派生行
  概述 245
配置
  代理程序和进程技术模型 32
  更改数据库参数 408
  内存堆 30
  文件系统高速缓存 163
  应用程序的 LDAP 用户 94
配置参数
  发现（DAS） 558
  联合 446
  描述 407
  内存参数之间的交互 23
  认证 431
  认证（DAS） 555
  使用配置顾问程序定义作用域 50
  数据库
    更改 407
  页大小 539
  影响查询优化 423
  影响代理程序数目 422
  用户出口 552
  摘要
    节标题描述 411
    数据库 411
    数据库管理器 411
  agentpri 428
  agent_stack_sz 426
  alt_collate 481
  appgroup_mem_sz 482
  applheapsz 484
  appl_memory 483
  app_ctl_heap_sz 481
  archretrydelay 485
  aslheapsz 429

配置参数 *(续)*
audit_buf_sz 430
autorestart 487
auto_del_rec_obj 485
auto_maint 485
avg_appls 488
backup_pending 488
blk_log_dsk_ful 489
blocknonlogged 489
catalogcache_sz 490
catalog_noauth 432
chngpgs_thresh 491
clnt_krb_plugin 433
clnt_pw_plugin 433
cluster_mgr 434
codepage 491
codeset 492
collate_info 492
comm_bandwidth 434
conn_elapse 435
contact_host 555
cpuspeed 435
dasadm_group 556
das_codepage 556
das_territory 556
database_consistent 493
database_level 493
database_memory 493
db2system 557
dbheap 496
db_mem_thresh 495
decflt_rounding 497
dftdbpath 437
dft_account_str 436
dft_degree 498
dft_extent_sz 499
dft_loadrec_ses 499
dft_monswitches 436
dft_mttb_types 500
dft_prefetch_sz 500
dft_queryopt 501
dft_refresh_age 502
dft_sqlmathwarn 502
diaglevel 438, 557
diagpath 439
dir_cache 442
discover 443
discover_db 503
discover_inst 443
dlchktime 504
dyn_query_mgmt 504
enable_xmlchar 505
exec_exp_task 558
failarchpath 505
fcm_num_buffers 444
fcm_num_channels 445

配置参数 *(续)*
federated_async 446
fed_noauth 446
fenced_pool 447
groupheap_ratio 506
group_plugin 448
hadr_db_role 506
hadr_local_host 506
hadr_local_svc 507
hadr_peer_window 507
hadr_remote_host 508
hadr_remote_inst 508
hadr_remote_svc 509
hadr_syncmode 509
hadr_timeout 510
health_mon 449
indexrec 449, 510
instance_memory 451
intra_parallel 453
java_heap_sz 453
jdk_64_path 512
jdk_path 454
jdk_path（DAS） 559
keepfenced 455
local_gssplugin 455
locklist 512
locktimeout 515
logarchmeth1 516
logarchmeth2 517
logarchopt1
  详细信息 518
logarchopt2
  详细信息 519
logbufsz 519
logfilsiz 520
loghead 521
logindexbuild 521
logpath 521
logprimary 522
logretain 523
logsecond 524
log_retain_status 516
maxagents 459
maxappls 525
maxcagents 460
maxfilop 526
maxlocks 526
maxlog 525
max_connections 456
  限制 425
max_connretries 456
max_coordagents 457
  限制 425
max_querydegree 458
max_time_diff 458
mincommit 529

配置参数 *(续)*
min_dec_div_3 528
mirrorlogpath 530
mon_heap_sz 461
multipage_alloc 531
newlogpath 531
nodetype 461
notifylevel 462
numarchretry 537
number_compat 538
numdb 465
numlogspan 536
numsegs 538
num_db_backups 532
num_freqvalues 533
num_initagents 463
num_initfenced 463
num_iocleaners 534
num_ioservers 535
num_poolagents 464
num_quantiles 536
overflowlogpath 538
pckcachesz 539
priv_mem_thresh 541
query_heap_sz 465
rec_his_retentn 541
release 466
restore_pending 542
restrict_access 542
resync_interval 467
rollfwd_pending 542
rqrioblk 467
sched_enable 559
sched_userid 560
self_tuning_mem 543
seqdetect 544
sheapthres 468
sheapthres_shr 545
smtp_server 560
softmax 546
sortheap 547
spm_log_file_sz 469
spm_log_path 470
spm_max_resync 470
spm_name 471
srvcon_auth 471
srvcon_gssplugin_list 472
srvcon_pw_plugin 472
srv_plugin_mode 473
start_stop_time 473
stat_heap_sz 548
stmtheap 549
svcename 474
sysadm_group 474
sysctrl_group 475
sysmaint_group 475
配置参数 *(续)*
sysmon_group 476
territory 550
tm_database 477
toolscat_db 560
toolscat_inst 561
toolscat_schema 561
tp_mon_name 477
trackmod 550
trust_allclnts 479
trust_clntauth 479
tsm_mgmtclass 550
tsm_nodename 551
tsm_owner 551
tsm_password 552
user_exit_status 552
util_heap_sz 553
util_impact_lim 480
varchar2_compat 553
vendoropt
详细信息 554
wlm_collect_int 配置参数 554
配置顾问程序
定义配置参数的作用域 50
关于 50
描述 17
生成建议的值 50
样本输出 51
配置文件
描述 407
位置 407
配置文件发行版级别配置参数 466

## [ Q ]

启动与停止超时配置参数 473
启用跟踪已修改的页配置参数 550
启用用户出口配置参数 552
迁移后任务
DB2 服务器
调整系统临时表空间页大小 151
前触发器 276
描述 276
与检查约束比较 251
前滚实用程序
前滚暂挂指示器 542
嵌套视图
定义 308
切换 DB2 副本 11
轻量级目录访问协议（LDAP）
安全性 71
编目节点项 92
创建用户 93
对象类和属性 72
更新协议信息 95
禁用 95

轻量级目录访问协议（LDAP） *(续)*
扩展目录模式 81
描述 71
目录服务 110
启用 90
设置注册表变量 94
刷新条目 96
搜索
目录分区 97
目录域 97
远程连接 96
支持 82
注册
数据库 92
主机数据库 82
DB2 服务器 91
注销
服务器 93
数据库 93
DB2 Connect 82
Windows 2003 Active Directory 89
全局级概要文件注册表 333
全局临时表
与其他表类型比较 205
全局（用户定义的）临时表
创建 231
全球标准时间 458
权限
定义组名
系统管理权限组名配置参数 474
系统控制权限组名配置参数 475
系统维护权限组名配置参数 475
缺省数据库路径配置参数 437
缺省值
压缩 222
缺省 SMS 容器数配置参数 538

## [ R ]

绕过联合认证配置参数 446
认证
可信客户机认证配置参数 479
信赖所有客户机配置参数 479
认证配置参数 431
认证 DAS 配置参数 555
日志
磁盘已满时阻止进行日志记录配置参数 489
辅助日志文件数配置参数 524
恢复范围和软检查点时间间隔配置参数 546
镜像日志路径配置参数 530
启用日志保留配置参数 523
启用用户出口配置参数 552
日志保留状态指示器配置参数 516
日志缓冲区大小配置参数 519
日志文件大小配置参数 520
日志文件的位置配置参数 521
日志 *(续)*
首个活动日志文件配置参数 521
溢出日志路径配置参数 538
原始设备 169
主日志文件数配置参数 522
newlogpath 配置参数 531
日志文件
空间需求 109
容量
扩展 3
容器
删除
调整预取大小 158
添加
调整预取大小 158
DMS 表空间 176
重新平衡和删除 177
减少容器，位于 185
删除容器，从 185
修改容器，位于 183

## [ S ]

删除规则
描述 245
设置完整性暂挂状态 245
深度压缩
*请参阅* 行压缩
生成的列
定义 211
示例 211
修改 235
升级之前锁定列表的最大百分比配置参数 526
声明 573
时间
节点之间的差异，最大值 458
死锁配置参数，检查的时间间隔 504
时间戳记
行更改 227
十进制除法，小数位为 3 的配置参数 528
实例
除去 68
创建 57
创建其他 61
多个 12
多个（Linux 和 UNIX） 60
多个（Windows） 12, 61
概述 12
更新配置
UNIX 63
Windows 64
目录 59
启动（Linux 和 UNIX） 65
启动（Windows） 65
缺省值 57, 59
缺省值，设置 11

实例 *(续)*
设计 58
设置当前的 339
使用 64
停止（Linux 和 UNIX） 66
停止（Windows） 67
同时运行 14, 66
修改 62
运行多个（Windows） 14
自动启动 64
instance_memory 配置参数 451
实例概要文件注册表 333
实例级概要文件注册表 333
实用程序操作
约束冲突 255
实用程序调速
关于 53
描述 17
视图
别名 237
不可用 312
创建 310
概述 305
恢复不起作用 312
可删除
使用 309
描述 305
嵌套视图的定义 308
删除 313
设计 306
使用检查选项
示例 306
使用用户定义的函数 311
修改 312
只读
使用 310
insertablex 310
updatablex 310
事务处理监视器
事务处理监视器名称配置参数 477
首个活动日志文件配置参数 521
受防护方式进程
运行供应商库函数 34
书籍
印刷版
订购 568
数据
表示 126
数据定义语言（DDL）
描述 101
语句
描述 101
数据访问优化
概述 18
数据分区
创建 231
数据服务器
概述 3
容量管理 3
数据库
备份
自动 17, 18
编目
概述 117
别名
创建 119
程序包依赖性 256
地域代码配置参数 492
发行版级别配置参数 466
分布式 101
分区 101
复原 115
估计大小要求 108
关系 101
配置参数摘要 411
配置, 跨多个分区 33
删除
DROP DATABASE 命令 127
设计
概述 101
同时处于活动状态的数据库的最大数目 465
整理信息 492
自动存储器 39, 112
appl_memory 配置参数 483
autorestart 配置参数 487
backup_pending 配置参数 488
codepage 配置参数 491
codeset 配置参数 492
territory 配置参数 550
数据库的备份次数配置参数 532
数据库地域代码配置参数 492
数据库堆配置参数 496
数据库对象
概述 203
命名规则
概述 318
NLS 320
Unicode 321
修改对象时的语句依赖性 256
数据库分区
编目
节点目录 105
概述 129
节点目录 105
数据库管理的空间（DMS）
表空间
创建 172
改变 176
容器（减小） 185
容器（删除） 185
与 SMS 表空间比较 152
表空间容器 177

数据库管理的空间（DMS） *(续)*
表空间映射 144
工作负载 153
描述 142
容器
重新平衡 177
删除 177
缩小大小 185
设备 153
数据库管理器
绑定实用程序 118
多个实例 12
机器节点类型配置参数 461
启动超时 473
停止超时 473
限制 323
数据库管理器配置参数
建议值 50
摘要 411
数据库恢复日志
在创建数据库期间分配 108
数据库目录
结构 102
数据库配置参数
建议值 50
数据库配置文件
创建 104
更改 107
数据库系统监视器
缺省数据库系统监视器开关配置参数 436
数据类型
缺省值 213
数据碎片整理
概述 18
数据组织方案
表分区 230
属性
列
更改 234
Netscape LDAP 84
双向索引 265
双字节字符集（DBCS）
命名规则 320
顺序值
生成 297
死锁
检查 504
dlchktime 配置参数 504
锁定
检查死锁的时间间隔配置参数 504
乐观锁定 223
升级之前锁定列表的最大百分比 526
锁定列表的最大存储量 512
锁定列表的最大存储量配置参数 512
索引
重建 272
索引 *(续)*
重命名 272
创建 271
非集群 265
非唯一 265
概述 263
集群 265
空间需求 268
描述 263
清除 53, 55
删除 273
设计 266
设计顾问程序 267
双向 265
提高性能 265
唯一 265
修改 272
延迟清除 55
异步清除 53, 55
用于设计的工具 267
准则和注意事项 266

## [ T ]

条款和条件
出版物的使用 572
通信
连接耗用时间 435
通知级别配置参数
概述 462
同时处于活动状态的数据库的最大数目配置参数 465
统计信息
自动收集 45, 49
统计信息概要分析
关于 18
脱机维护 19

## [ W ]

外键
引用完整性的隐含意义 255
用于定义的规则 251
与引用约束交互 255
约束 245
约束名称 251
组合 251
外键约束 243
引用完整性规则 245
引用约束 251
用于定义的规则 251
维护
时间段 19
自动 18
维护时间段
自动 19

唯一键
描述 245
使用序列来生成 293
唯一索引 265
唯一约束 243, 244
定义 245
描述 213
设计 249
文档
概述 565
使用条款和条件 572
印刷版 565
PDF 565
文件系统
为表空间高速缓存 160, 163
问题确定
教程 572
可用的信息 572

## [ X ]

系统管理的空间（SMS）
表空间
描述 140
系统临时表空间 172
页大小
DB2 服务器的迁移后任务 151
系统目录视图
描述 306
系统时钟
更改注意事项 227
系统数据库目录
查看 127
描述 105
限制
标识长度 323
命名规则 317
自动存储器 42, 114
SQL 323
向导
配置顾问程序 107
协议
TCP/IP 服务名称配置参数 474
性能
表空间 188
管理序列 294
使用索引来提高性能 265
性能配置向导
重命名为“配置顾问程序” 107
序列
查看 298
创建 296
管理行为 294
恢复使用序列的数据库 296
删除 299
设计 293
序列 *(续)*
生成 293
使用 297
示例 299
修改 298
应用程序性能 295
与标识列比较 295, 297
值 300
序列表达式
描述 297

## [ Y ]

压缩
表
行 44, 218
列值 222
缺省系统值 217, 222
数据行
详细信息 44, 217, 218
值 217, 222
NULL 222
压缩字典
复位 42, 219
行压缩中的角色 44, 218
描述 219
手动创建 42, 219
自动创建 42, 219
延迟索引清除
监视 55
页
大小
表空间 167
数据库缺省 539
页大小
表 215
引用完整性
插入规则 245
更新规则 245
描述 213
删除规则 245
约束 245
引用约束
定义 251
描述 245
与外键交互 255
PRIMARY KEY 子句, CREATE/ALTER TABLE 语句 251
REFERENCES 子句, CREATE/ALTER TABLE 语句 251
应用程序
节点的最大协调代理程序数 457
控制堆, 设置 481
控制序列 294
请求器 120
应用程序导向的分布式工作单元 123
应用程序进程
连接状态 124

应用程序控制堆大小配置参数 481
应用程序性能
  序列 295
  序列与标识列的比较 295
应用程序支持层堆大小配置参数 429
应用程序组内存集的最大大小配置参数 482
用户标识
  命名规则 320
用户表页限制 216
用户出口数据库配置参数 552
用户出口状态指示器配置参数 552
用户定义的单值数据类型 213
用户定义的函数
  与视图配合使用 311
用户定义的（全局）临时表
  创建 231
用户临时表空间
  创建 172
用户数据
  目录 439
语句堆大小配置参数 549
预取大小
  手动调整
    含义 158
  自动调整 158
源表
  创建 231
原始日志 169
原始设备 172
原始 I/O
  在 Linux 上设置 171
  指定 169
远程工作单元
  分布式关系数据库 120
约束
  表检查 245
  （表）检查 245
  参考 245, 249, 256
  查看表的定义 259
  创建 258
  定义
    外键 251
    引用约束 251
  类型 243
  描述 243
  删除 260
  设计 249
    检查约束 250
  唯一 245
  唯一（键） 244
  修改 258
  引用 245
  与前触发器比较 251
  与外键交互 255
  主键 245
  NOT NULL 244
运行状况监视器
  描述 17
  health_mon 配置参数 449

## [ Z ]

摘要表
  恢复不起作用 236
  与其他表类型比较 205
只读视图
  使用 310
值
  序列 300
值压缩 222
主键
  描述 213, 245
  设计 250
主键约束
  概述 243
注册表变量
  概述 341
  环境变量 333
  聚集 340
  声明 335
  注册表变量
    DB2_HADR_PEER_WAIT_ LIMIT 389
    DB2_LOGGER_NON_ BUFFERED_IO 372
  DB2ACCOUNT 346
  DB2ADMINSERVER 389
  DB2ASSUMEUPDATE 372
  DB2BIDI 346
  DB2BPVARS 372
  DB2BQTIME 364
  DB2BQTRY 364
  DB2CHECKCLIENTINTERVAL 362
  DB2CHGPWD_ESE 366
  DB2CHKPTR 372
  DB2CHKSQLDA 372
  DB2CLIINIPATH 389
  DB2CODEPAGE 346
  DB2COMM 362
  DB2CONNECT_DISCONNECT_ON_INTERRUPT 389
  DB2CONNECT_ENABLE_EURO_CODEPAGE 354
  DB2CONNECT_IN_APP_PROCESS 354
  DB2CONSOLECP 346
  DB2COUNTRY 346
  DB2DBDFT 346
  DB2DBMSADDR 346
  DB2DEFPREP 389
  DB2DISCOVERYTIME 346
  DB2DMNBCKCTLR 389
  DB2DOMAINLIST 354
  DB2ENVLIST 354
  DB2FCMCOMM 362
  DB2FODC 346
  DB2GRAPHICUNICODESERVER 346

注册表变量 *(续)*
DB2INCLUDE 346
DB2INSTANCE 354
DB2INSTDEF 346
DB2INSTOWNER 346
DB2INSTPROF 354
DB2IQTIME 364
DB2LDAPCACHE 389
DB2LDAPHOST 389
DB2LDAPSecurityConfig 354
DB2LDAP_BASEDN 389
DB2LDAP_CLIENT_PROVIDER 389
DB2LDAP_KEEP_CONNECTION 389
DB2LDAP_SEARCH_SCOPE 389
DB2LIBPATH 354
DB2LOADREC 389
DB2LOCALE 346
DB2LOCK_TO_RB 389
DB2LOGINRESTRICTIONS 354
DB2MAXFSCRSEARCH 372
DB2MEMDISCLAIM 372
DB2MEMMAXFREE 372
DB2NODE 354
DB2NOEXITLIST 389
DB2NTMEMSIZE 372
DB2NTNOCACHE 372
DB2NTPRICLASS 372
DB2NTWORKSET 372
DB2OPTIONS 354
DB2PATH 354
DB2PORTRANGE 366
DB2PRIORITIES 372
DB2PROCESSORS 354
DB2RCMD_LEGACY_MODE 354
DB2REMOTEPREG 389
DB2ROUTINE_DEBUG 389
DB2RQTIME 364
DB2RSHCMD 362
DB2RSHTIMEOUT 362
DB2SATELLITEID 389
DB2SLOGON 346
DB2SORCVBUF 362
DB2SORT 389
DB2SOSNDBUF 362
DB2SYSTEM 354
DB2TCPCONNMGRS 362
DB2TCP_CLIENT_CONTIMEOUT 362
DB2TCP_CLIENT_RCVTIMEOUT 362
DB2TERRITORY 346
DB2_ALLOCATION_SIZE 372
DB2_ALTERNATE_GROUP_LOOKUP 354
DB2_ANTIJOIN 368
DB2_APM_PERFORMANCE 372
DB2_ASYNC_IO_MAXFILOP 372
DB2_ATS_ENABLE 389
DB2_AVOID_PREFETCH 372

注册表变量 *(续)*
DB2_CAPTURE_LOCKTIMEOUT 346
DB2_CLPHISTSIZE 364
DB2_CLPPROMPT 364
DB2_CLP_EDITOR 364
DB2_COLLECT_TS_REC_INFO 346
DB2_COMMIT_ON_EXIT 389
DB2_CONNRETRIES_INTERVAL 346
DB2_COPY_NAME 354
DB2_CREATE_DB_ON_PATH 389
DB2_DIAGPATH 354
DB2_DISABLE_FLUSH_LOG 389
DB2_DISPATCHER_PEEKTIMEOUT 389
DB2_DJ_INI 389
DB2_DOCHOST 389
DB2_DOCPORT 389
DB2_ENABLE_AUTOCONFIG_DEFAULT 389
DB2_ENABLE_LDAP 389
DB2_EVALUNCOMMITTED 372
DB2_EVMON_EVENT_LIST_SIZE 389
DB2_EVMON_STMT_FILTER 389
DB2_EXTENDED_IO_FEATURES 372
DB2_EXTENDED_OPTIMIZATION 372
DB2_EXTSECURITY 389
DB2_FALLBACK 389
DB2_FCM_SETTINGS 366
DB2_FMP_COMM_HEAPSZ 389
DB2_FORCE_APP_ON_MAX_LOG 346
DB2_FORCE_NLS_CACHE 362
DB2_GRP_LOOKUP 389
DB2_HADR_BUF_SIZE 389
DB2_HADR_NO_IP_CHECK 389
DB2_HADR_SORCVBUF 389
DB2_HADR_SOSNDBUF 389
DB2_HASH_JOIN 372
DB2_INLIST_TO_NLJN 368
DB2_IO_PRIORITY_SETTING 372
DB2_KEEPTABLELOCK 372
DB2_KEEP_AS_AND_DMS_ CONTAINERS_OPEN 372
DB2_LARGE_PAGE_MEM 372
DB2_LIC_STAT_SIZE 346
DB2_LIKE_VARCHAR 368
DB2_LOAD_COPY_NO_OVERRIDE 389
DB2_MAP_XML_AS_CLOB_FOR_ DLC 389
DB2_MAX_CLIENT_CONNRETRIES 346
DB2_MAX_INACT_STMTS 372
DB2_MAX_LOB_BLOCK_SIZE 389
DB2_MAX_NON_TABLE_LOCKS 372
DB2_MDC_ROLLOUT 372
DB2_MEMORY_PROTECT 389
DB2_MEM_TUNING_RANGE 372
DB2_MINIMIZE_LISTPREFETCH 368
DB2_MMAP_READ 372
DB2_MMAP_WRITE 372
DB2_NEW_CORR_SQ_FF 368
DB2_NO_FORK_CHECK 372

注册表变量 *(续)*
DB2_NUM_CKPW_DAEMONS 389
DB2_NUM_FAILOVER_NODES 366
DB2_OBJECT_TABLE_ENTRIES 372
DB2_OPTSTATS_LOG 389
DB2_OPT_MAX_TEMP_SIZE 368
DB2_OVERRIDE_BPF 372
DB2_PARALLEL_IO 354
DB2_PARTITIONEDLOAD_ DEFAULT 366
DB2_PINNED_BP 372
DB2_REDUCED_ OPTIMIZATION 368
DB2_RESOLVE_CALL_ CONFLICT 389
DB2_RESOURCE_POLICY 372
DB2_SELECTIVITY 368
DB2_SELUDI_COMM_BUFFER 372
DB2_SERVER_CONTIMEOUT 389
DB2_SERVER_ENCALG 389
DB2_SET_MAX_CONTAINER_ SIZE 372
DB2_SKIPDELETED 372
DB2_SKIPINSERTED 372
DB2_SMS_TRUNC_TMPTABLE_ THRESH 372
DB2_SORT_AFTER_TQ 372
DB2_SQLROUTINE_PREPOPTS 368
DB2_SYSTEM_MONITOR_ SETTINGS 346
DB2_THREAD_SUSPENSION 389
DB2_TRUNCATE_ REUSESTORAGE 389
DB2_TRUSTED_BINDIN 372
DB2_UPDDBCFG_SINGLE_DBPARTITION 354
DB2_USE_ALTERNATE_PAGE_ CLEANING 372
DB2_USE_DB2JCCT2_ JROUTINE 389
DB2_USE_FAST_PREALLOCATION 372
DB2_USE_PAGE_CONTAINER_TAG 354
DB2_UTIL_MSGPATH 389
DB2_VENDOR_INI 389
DB2_VIEW_REOPT_VALUES 346
DB2_WORKLOAD 354
DB2_XBSA_LIBRARY 389
NO_SORT_MGJOIN 关键字 368
NO_SORT_NLJOIN 关键字 368
转出
延迟清除 55
转换变量
使用触发器, 访问旧列值和新列值 285
转换表
用于触发器，引用旧表结果集和新表结果集 286
追加方式表
与其他表类型比较 205
字典
行压缩 219
自调整内存
分区数据库环境 27, 28
概述 20
监视 26
禁用 26
局限性 23
描述 17
自调整内存 *(续)*
启用 25, 543
非统一环境 28
自动
调整表空间大小 36, 155
调整预取大小
在添加或删除容器之后 158
自动重新启动启用配置参数 487
自动创建字典
详细信息 42, 219
自动存储路径
添加 42, 114
自动存储器
表空间 34, 148
关于 34
描述 17
适用于数据库 39, 112
限制 42, 114
自动存储器表空间
改变 187
自动调整内存 26
自动功能 17
在缺省情况下启用 17
自动配置 API 50
自动收集统计信息 49
描述 17
自动维护
关于 18
时间段 19
字符串
数据类型
零长度 213
字符串行设备 172
自引用表 245
自引用行 245
自主计算
关于 17
组
命名规则 320
最大并发代理程序数配置参数 460
最大查询并行度配置参数 458
对查询优化的影响 423
最大代理程序数配置参数 459
最大活动应用程序数配置参数 525
最大受防护的进程数配置参数 447
最大协调代理程序数配置参数 457
最大 Java 解释器堆大小配置参数 453
最先合适顺序 216

# A

Active Directory
安全性 88
扩展目录模式 89
配置 DB2 88
轻量级目录访问协议（LDAP） 71

Active Directory *(续)*
支持 88
DB2 对象 89
ADC
*请参阅* 自动创建字典
ADMIN_COPY_SCHEMA 过程
模式复制的示例 197
agentpri 数据库管理器配置参数 428
agent_stack_sz 数据库管理器配置参数 426
AIX
大内存页支持 4
固定共享内存 5
系统命令
vmo 4, 5
ALTER 触发器
描述 276
ALTER COLUMN 子句
在表列中 235
ALTER TABLESPACE 语句
示例 176
alt_collate 配置参数 481
appgroup_mem_sz 数据库管理器配置参数 482
appl_memory 配置参数
内存参数之间的交互 23
appl_memory 数据库配置参数 483
app_ctl_heap_sz 数据库配置参数 481
archretrydelay 配置参数 485
aslheapsz 配置参数 429
ATTACH 命令 66
audit_buf_sz 配置参数 430
AUTHID 标识
限制 317
AUTOCONFIGURE 命令 50
样本输出 51
auto_del_rec_obj 数据库配置参数 485
auto_maint 配置参数 485
avg_appls 配置参数 488

# B

backup_pending 配置参数 488
BEFORE DELETE 触发器
描述 276
blk_log_dsk_ful 配置参数
详细信息 489
blocknonlogged 配置参数 489

# C

CATALOG DATABASE 命令
示例 117
catalogcache_sz 数据库配置参数 490
catalog_noauth 配置参数 432
chngpgs_thresh 配置参数 491
CIO/DIO
用作缺省值 162
clnt_krb_plugin 配置参数 433
clnt_pw_plugin 配置参数 433
cluster_mgr 配置参数 434
codepage 数据库配置参数 491
codeset 数据库配置参数 492
collate_info 数据库配置参数 492
comm_bandwidth 数据库管理器配置参数
对查询优化的影响 423
描述 434
conn_elapse 配置参数 435
contact_host 配置参数 555
cpuspeed 配置参数
对查询优化的影响 423
描述 435
CREATE DATABASE 命令
示例 110
CREATE TABLE 语句
定义引用约束 251
CREATE TABLESPACE 语句
调整系统临时表空间页大小 151
CURRENT SCHEMA 专用寄存器 194

# D

DAS 发现方式配置参数 558
dasadm_group 配置参数 556
das_codepage 配置参数 556
das_territory 配置参数 556
database_consistent 配置参数 493
database_level 配置参数 493
database_memory 配置参数
内存参数之间的交互 23
database_memory 数据库配置参数
描述 493
自调整 20
DATE 数据类型
缺省值 213
DB2 服务器
迁移后任务
调整系统临时表空间页大小 151
DB2 副本
更新 13
缺省 IBM 数据库客户机接口副本 7
同一台计算机上的多个 DB2 副本
缺省实例设置 11
DB2 管理服务器（DAS）设置 10
DB2 管理服务器（DAS）
多个 DB2 副本设置 10
配置参数
认证 555
contact_host 555
dasadm_group 556
das_codepage 556
das_territory 556

DB2 管理服务器（DAS） *(续)*
配置参数 *(续)*
db2system 557
exec_exp_task 558
jdk_64_path 512
jdk_path 559
sched_enable 559
sched_userid 560
smtp_server 560
toolscat_db 560
toolscat_inst 561
toolscat_schema 561
所有权规则 338
DB2 信息中心
版本 569
查看各种语言版本 569
更新 570
语言 569
DB2ACCOUNT 注册表变量
描述 346
DB2ADMINSERVER 变量 389
DB2ASSUMEUPDATE 注册表变量 372
DB2AUTH
注册表变量 389
DB2BIDI 注册表变量
描述 346
DB2BPVARS 注册表变量
描述 372
DB2BQTIME 注册表变量 364
DB2BQTRY 注册表变量 364
DB2CHECKCLIENTINTERVAL 变量 362
DB2CHGPWD_EEE 注册表变量 366
DB2CHKPTR 变量 372
DB2CHKSQLDA 变量 372
DB2CLIINIPATH 变量
描述 389
DB2CODEPAGE 注册表变量
描述 346
DB2COMM 变量 362
DB2CONNECT_DISCONNECT_ON_INTERRUPT 变量 389
DB2CONNECT_ENABLE_EURO_CODEPAGE 354
DB2CONNECT_IN_APP_PROCESS 环境变量 354
DB2CONSOLECP 注册表变量 346
DB2COUNTRY 注册表变量
描述 346
DB2DBDFT 注册表变量 346
DB2DBMSADDR 注册表变量 346
DB2DEFPREP 注册表变量
描述 389
DB2DISCOVERYTIME 注册表变量 346
DB2DMNBCKCTLR 注册表变量
描述 389
DB2DOMAINLIST 变量
描述 354
db2envar.bat 命令
用来切换 DB2 副本 11
DB2ENVLIST 环境变量 354
DB2FCMCOMM 变量 362
DB2FODC 注册表变量
描述 346
DB2GRAPHICUNICODESERVER 注册表变量
描述 346
db2icrt 命令
创建实例 61
db2idrop 命令
删除实例 68
DB2INCLUDE 注册表变量 346
DB2INSTANCE
设置 11
DB2INSTANCE 环境变量
定义缺省实例 12
描述 354
DB2INSTDEF
设置 11
DB2INSTDEF 注册表变量 346
DB2INSTOWNER 注册表变量 346
DB2INSTPROF 注册表变量
描述 354
位置 407
DB2IQTIME 注册表变量 364
db2iupdt 命令
更新实例配置
Linux 63
UNIX 63
Windows 64
DB2LDAPCACHE 变量 389
DB2LDAPHOST 变量
描述 389
DB2LDAPSecurityConfig 环境变量 354
DB2LDAP_BASEDN 变量
描述 389
DB2LDAP_CLIENT_PROVIDER 注册表变量
描述 389
IBM LDAP 客户机 82
DB2LDAP_KEEP_CONNECTION 注册表变量
描述 389
DB2LDAP_SEARCH_SCOPE 变量
描述 389
db2ldcfg 命令
配置 LDAP 用户 94
DB2LIBPATH 环境变量 354
DB2LOADREC 注册表变量
描述 389
DB2LOCALE 注册表变量
描述 346
DB2LOCK_TO_RB 变量 389
DB2LOGINRESTRICTIONS 变量 354
DB2MAXFSCRSEARCH 变量 372
DB2MEMDISCLAIM 注册表变量 372
DB2MEMMAXFREE 注册表变量
描述 372

db2move 命令
  模式复制示例 198
  COPY 模式错误 199
DB2NODE 环境变量
  描述 354
db2nodes.cfg 文件
  创建 106
  概述 59
DB2NOEXITLIST 注册表变量
  描述 389
DB2NTMEMSIZE 变量 372
DB2NTNOCACHE 注册表变量
  描述 372
  NO FILE SYSTEM CACHING 子句比较 160
DB2NTPRICLASS 注册表变量
  描述 372
DB2NTWORKSET 变量 372
DB2OPTIONS 环境变量
  描述 354
DB2PATH 环境变量 354
DB2PORTRANGE 注册表变量 366
DB2PRIORITIES 注册表变量
  描述 372
DB2PROCESSORS 环境变量 354
DB2RCMD_LEGACY_MODE 环境变量 354
DB2REMOTEPREG 变量 389
DB2ROUTINE_DEBUG 注册表变量 389
DB2RQTIME 注册表变量 364
DB2RSHCMD 注册表变量 362
DB2RSHTIMEOUT 注册表变量 362
DB2SATELLITEID 变量 389
db2SelectDB2Copy API
  用来切换 DB2 副本 11
db2set 命令
  管理注册表变量和环境变量 333, 335
DB2SORCVBUF 变量
  描述 362
DB2SORT 变量 389
DB2SOSNDBUF 变量
  描述 362
DB2SYSTEM 环境变量 354
db2system 配置参数 557
DB2TCPCONNMGRS 注册表变量 362
DB2TCP_CLIENT_CONTIMEOUT 注册表变量 362
DB2TCP_CLIENT_RCVTIMEOUT 注册表变量
  描述 362
DB2TERRITORY 注册表变量
  描述 346
DB2_ALLOCATION_SIZE 注册表变量
  描述 372
DB2_ALTERNATE_GROUP_LOOKUP 环境变量 354
DB2_ANTIJOIN 变量 368
DB2_APM_PERFORMANCE 变量 372
DB2_ASYNC_IO_MAXFILOP 注册表变量
  描述 372
DB2_ATS_ENABLE 注册表变量 389
DB2_AVOID_PREFETCH 变量 372
DB2_CAPTURE_LOCKTIMEOUT 注册表变量
  描述 346
DB2_CLPHISTSIZE 注册表变量 364
DB2_CLPPROMPT 注册表变量 364
DB2_CLP_EDITOR 注册表变量 364
DB2_COLLECT_TS_REC_INFO 注册表变量 346
DB2_COMMIT_ON_EXIT 注册表变量 389
DB2_CONNRETRIES_INTERVAL 注册表变量
  描述 346
DB2_COPY_NAME 环境变量 354
DB2_CREATE_DB_ON_PATH 注册表变量 389
DB2_DIAGPATH 变量
  描述 354
DB2_DISABLE_FLUSH_LOG 注册表变量 389
DB2_DISPATCHER_PEEKTIMEOUT 注册表变量 389
DB2_DJ_INI 变量 389
DB2_DOCHOST 变量 389
DB2_DOCPORT 变量 389
DB2_ENABLE_AUTOCONFIG_DEFAULT 变量 389
DB2_ENABLE_LDAP 变量
  描述 389
DB2_EVALUNCOMMITTED 注册表变量
  描述 372
DB2_EVMON_EVENT_LIST_SIZE 注册表变量
  描述 389
DB2_EVMON_STMT_FILTER 注册表变量 389
DB2_EXTENDED_IN2JOIN 变量 372
DB2_EXTENDED_IO_FEATURES 变量
  描述 372
DB2_EXTENDED_OPTIMIZATION 变量 372
DB2_EXTSECURITY 注册表变量 389
DB2_FALLBACK 变量 389
DB2_FMP_COMM_HEAPSZ 变量 389
DB2_FORCE_APP_ON_MAX_LOG 注册表变量 346
DB2_FORCE_NLS_CACHE 注册表变量
  描述 362
DB2_GRP_LOOKUP 变量 389
DB2_HADR_BUF_SIZE 变量 389
DB2_HADR_NO_IP_CHECK 变量 389
DB2_HADR_SORCVBUF 注册表变量 389
DB2_HADR_SOSNDBUF 注册表变量 389
DB2_HASH_JOIN 注册表变量
  描述 372
DB2_INLIST_TO_NLJN 注册表变量 368
DB2_IO_PRIORITY_SETTING 注册表变量 372
DB2_KEEPTABLELOCK 注册表变量 372
DB2_LARGE_PAGE_MEM 注册表变量
  描述 372
DB2_LIC_STAT_SIZE 注册表变量 346
DB2_LIKE_VARCHAR 注册表变量 368
DB2_LOAD_COPY_NO_OVERRIDE 变量 389
DB2_MAP_XML_AS_CLOB_FOR_DLC 注册表变量
  描述 389
DB2_MAX_CLIENT_CONNRETRIES 注册表变量
  描述 346

DB2_MAX_INACT_STMTS 变量 372
DB2_MAX_LOB_BLOCK_SIZE 变量 389
DB2_MAX_NON_TABLE_LOCKS 变量 372
DB2_MDC_ROLLOUT 注册表变量
描述 372
DB2_MEMORY_PROTECT 注册表变量
描述 389
DB2_MEM_TUNING_RANGE 变量 372
DB2_MINIMIZE_LISTPREFETCH 注册表变量 368
DB2_MMAP_READ 变量 372
DB2_MMAP_WRITE 变量 372
DB2_NEW_CORR_SQ_FF 变量 368
DB2_NO_FORK_CHECK 注册表变量
描述 372
DB2_NUM_CKPW_DAEMONS 注册表变量 389
DB2_NUM_FAILOVER_NODES 注册表变量 366
DB2_OPTSTATS_LOG 注册表变量
描述 389
DB2_OPT_MAX_TEMP_SIZE 变量 368
DB2_OVERRIDE_BPF 变量 372
DB2_PARALLEL_IO 注册表变量
描述 188, 354
使用 158
DB2_PARTITIONEDLOAD_ DEFAULT 注册表变量
描述 366
DB2_PINNED_BP 注册表变量
描述 372
DB2_REDUCED_OPTIMIZATION 注册表变量 368
DB2_RESOLVE_CALL_CONFLICT 注册表变量 389
DB2_RESOURCE_POLICY 注册表变量
描述 372
DB2_SELECTIVITY 注册表变量 368
DB2_SELUDI_COMM_BUFFER 注册表变量 372
DB2_SERVER_CONTIMEOUT 注册表变量 389
DB2_SERVER_ENCALG 注册表变量 389
DB2_SET_MAX_CONTAINER_SIZE 注册表变量
描述 372
DB2_SKIPDELETED 注册表变量 372
DB2_SKIPINSERTED 注册表变量 372
DB2_SMS_TRUNC_TMPTABLE_THRESH 变量 372
DB2_SORT_AFTER_TQ 变量 372
DB2_SQLROUTINE_PREPOPTS 注册表变量 368
DB2_SYSTEM_MONITOR_SETTINGS 注册表变量
描述 346
DB2_THREAD_SUSPENSION 注册表变量
描述 389
DB2_TRUNCATE_REUSESTORAGE 注册表变量 389
DB2_TRUSTED_BINDIN 注册表变量
描述 372
DB2_UPDDBCFG_SINGLE_DBPARTITION 变量
描述 354
DB2_USE_ALTERNATE_PAGE_ CLEANING 注册表变量
描述 372
DB2_USE_DB2JCCT2_JROUTINE 变量
描述 389
DB2_USE_FAST_PREALLOCATION 注册表变量 372
DB2_USE_PAGE_CONTAINER_TAG 变量
对性能的影响 188
描述 354
DB2_UTIL_MSGPATH 注册表变量 389
DB2_VENDOR_INI 注册表变量
描述 389
DB2_VIEW_REOPT_VALUES 注册表变量 346
DB2_WORKLOAD 聚集注册表变量
描述 340, 354
DB2_XBSA_LIBRARY 注册表变量 389
DBCS（双字节字符集）
请参阅双字节字符集（DBCS） 320
dbheap 数据库配置参数
概述 496
db_mem_thresh 配置参数 495
DDL（数据定义语言）
请参阅“数据定义语言（DDL）” 101
decflt_rounding 数据库配置参数 497
DECLARE GLOBAL TEMPORARY TABLE 语句
概述 231
DETACH 命令
从实例拆离 66
dftdbpath 配置参数 437
dft_account_str 配置参数 436
dft_degree 配置参数
对查询优化的影响 423
描述 498
dft_extent_sz 配置参数 499
dft_loadrec_ses 配置参数 499
dft_monswitches 配置参数 436
dft_mon_bufpool 配置参数 436
dft_mon_lock 配置参数 436
dft_mon_sort 配置参数 436
dft_mon_stmt 配置参数 436
dft_mon_table 配置参数 436
dft_mon_timestamp 配置参数 436
dft_mon_uow 配置参数 436
dft_mttb_types 配置参数 500
dft_prefetch_sz 配置参数 500
dft_queryopt 配置参数 501
dft_refresh_age 配置参数 502
dft_sqlmathwarn 配置参数 502
diaglevel 配置参数 557
描述 438
diagpath 配置参数 439
dir_cache 配置参数 442
discover_db 配置参数 503
discover_inst 配置参数 443
dlchktime 配置参数 504
DMS（数据库管理空间）
请参阅“数据库管理空间（DMS）” 142
dyn_query_mgmt 配置参数
描述 504

## E

enable_xmlchar 数据库配置参数
  描述 505
exec_exp_task 配置参数 558

## F

failarchpath 配置参数 505
FCM（快速通信管理器）
  监视元素
    fcm_num_channels 445
  通道 445
fcm_num_buffers 配置参数 444
fcm_num_channel 配置参数 445
federated_async 数据库管理器配置参数 446
fed_noauth 配置参数 446
fenced_pool 数据库管理器配置参数 447
FILE SYSTEM CACHING 子句 160

## G

groupheap_ratio 数据库管理器配置参数 506
group_plugin 配置参数 448

## H

hadr_db_role 配置参数 506
hadr_local_host 配置参数 506
hadr_local_svc 配置参数 507
hadr_peer_window 数据库配置参数
  描述 507
hadr_remote_host 配置参数 508
hadr_remote_inst 配置参数 508
hadr_remote_svc 配置参数 509
hadr_syncmode 配置参数 509
hadr_timeout 配置参数 510
health_mon 配置参数 449

## I

IBM 数据库客户机接口副本
  缺省值 7
IBM eNetwork Directory
  对象类和属性 72
IBM SecureWay Directory Server
  扩展目录模式，用于 83
IMPLICIT_SCHEMA 权限 191
indexrec 配置参数 449, 510
instance_memory 配置参数
  内存参数之间的交互 23
INSTEAD OF 触发器 277
  描述 276
intra_parallel 数据库管理器配置参数 453
I/O
  表空间注意事项 167
  并行性
    使用 RAID 设备 188

## J

java_heap_sz 数据库管理器配置参数 453
jdk_64_path 配置参数 512
jdk_path 配置参数
  描述 454
jdk_path DAS 配置参数 559

## K

keepfenced 配置参数
  描述 455

## L

LBAC（基于标号的访问控制）
  安全标号
    名称长度 323
    组件名称程度 323
  安全策略
    名称长度 323
  乐观锁定 224
  限制 323
LDAP（轻量级目录访问协议）
  安全性 71
  编目节点项 92
  重新路由客户机 95
  对象类和属性 72
  更新协议信息 95
  禁用 95
  扩展目录模式 81
  描述 71
  目录服务 110
  启用 90
  设置注册表变量 94
  刷新条目 96
  搜索
    目录分区 97
    目录域 97
  用户创建 93
  远程连接 96
  支持 82
  注册
    数据库 92
    主机数据库 82
    DB2 服务器 91
  注销
    服务器 93
    数据库 93
  DB2 Connect 82

LDAP（轻量级目录访问协议） *(续)*
Windows 2003 Active Directory 89
LOB（大对象）
高速缓存行为 153
local_gssplugin 配置参数 455
locklist 配置参数
查询优化 423
描述 512
locktimeout 配置参数 515
logarchmeth1 配置参数 516
logarchmeth2 配置参数 517
logarchopt1 配置参数
详细信息 518
logarchopt2 配置参数
详细信息 519
LOGBUFSZ 配置参数 519
logfilsiz 配置参数 520
loghead 配置参数 521
logindexbuild 配置参数 521
logpath 配置参数 521
logprimary 配置参数 522
logretain 数据库配置参数 523
logsecond 配置参数 524
log_retain_status 配置参数 516
LONG 数据
高速缓存行为 153

# M

maxagents 数据库管理器配置参数 459
maxappls 配置参数 525
对内存使用的影响 20
maxcagents 数据库管理器配置参数 460
maxcoordagents 配置参数 20
MAXDARI 配置参数
重命名为 fenced_pool 配置参数 447
maxfilop 数据库配置参数 526
maxlocks 配置参数 526
maxlog 配置参数 525
max_connections 数据库管理器配置参数
限制 425
max_connretries 配置参数 456
max_coordagents 数据库管理器配置参数 457
限制 425
max_logicagents 配置参数 456
max_querydegree 配置参数 458
max_time_diff 配置参数 458
mincommit 配置参数 529
min_dec_div_3 配置参数 528
mirrorlogpath 配置参数 530
mon_heap_sz 数据库管理器配置参数 461
MQT（具体化查询表）
改变属性 233
刷新数据 234
multipage_alloc 配置参数 531

# N

Netscape
LDAP 目录支持 84
newlogpath 配置参数 531
NEXT VALUE 表达式
使用标识列 297
序列 293
NO FILE SYSTEM CACHING 子句 160
nodetype 配置参数 461
NOT NULL 约束
概述 244
类型 243
NO_SORT_MGJOIN 368
NO_SORT_NLJOIN 368
NULL 数据类型 213
numarchretry 配置参数 537
number_compat 数据库配置参数
详细信息 538
NUMDB
配置参数 465
对内存使用的影响 20
numinitagents 配置参数 463
numlogspan 配置参数 536
numsegs 数据库配置参数
概述 538
num_db_backups 配置参数
概述 532
num_freqvalues 配置参数 533
num_initfenced 数据库管理器配置参数 463
num_iocleaners 配置参数 534
num_ioservers 配置参数 535
num_poolagents 数据库管理器配置参数 464
num_quantiles 配置参数 536

# O

OLTP（联机事务处理）
表空间设计 153
overflowlogpath 配置参数 538

# P

pagesize 配置参数 539
pckcachesz 配置参数 539
PREVIOUS VALUE 表达式
标识列 297
概述 293
priv_mem_thresh 数据库管理器配置参数
描述 541

# Q

query_heap_sz 数据库管理器配置参数 465

# R


RAID（独立磁盘冗余阵列）设备
  优化表空间性能 188
rec_his_retentn 配置参数 541
release 配置参数 466
Reorg 实用程序
  绑定至数据库 118
restore_pending 配置参数 542
RESTRICTIVE 选项
  CREATE DATABASE
    数据库配置参数 542
restrict_access 配置参数 542
resync_interval 配置参数 467
RID_BIT() 和 RID()
  内置函数 227
RID_BIT() 和 RID() 内置函数 227
RID_BIT() 内置函数 225
rollfwd_pending 配置参数 542
ROW CHANGE TIMESTAMP 列 225
rqrioblk 配置参数 467
RUNSTATS 命令
  自动收集统计信息 45, 49


# S


sched_enable 配置参数 559
sched_userid 配置参数 560
scope
  添加 235
self_tuning_mem
  配置参数 543
seqdetect 配置参数 544
sheapthres 配置参数 468
sheapthres_shr 配置参数 545
SMS（系统管理的空间）
  表空间
    创建 172
    描述 140
    与 DMS 表空间比较 152
  工作负载注意事项 153
  设备注意事项 153
SMS 表空间
  改变 176
SMS 目录
  在非自动存储器数据库中 102
smtp_server 配置参数 560
softmax 配置参数 546
sortheap 数据库配置参数
  对查询优化的影响 423
  描述 547
spm_log_file_sz 配置参数 469
spm_log_path 配置参数 470
spm_max_resync 配置参数 470
spm_name 配置参数 471
SQL（结构化查询语言）
  限制 323
SQL 语句
  不可用 256
  显示帮助 569
  优化
    配置参数 423
  语句堆大小配置参数 549
SQL PL 语句
  在触发器操作中受支持 284
SQLDBCON 配置文件 407, 408
SQLDBCON 数据库配置文件 104
SQLDBCONF 配置文件 407, 408
SQLDBCONF 数据库配置文件 104
srvcon_auth 配置参数 471
srvcon_gssplugin_list 配置参数 472
srvcon_pw_plugin 配置参数 472
srv_plugin_mode 配置参数 473
start_stop_time 配置参数 473
stat_heap_sz 数据库配置参数 548
STMM（自调整内存管理器）
  监视 26
  局限性 23
  启用 25, 543
stmtheap 数据库配置参数
  对查询优化的影响 423
  详细信息 549
Sun One Directory Server
  扩展目录模式，用于 86
svcename 配置参数 474
SWITCH ONLINE 子句 188
sysadm_group 配置参数 474
SYSCATSPACE 表空间 168
SYSCAT.INDEXES 视图
  查看表的约束定义 259
sysctrl_group 配置参数 475
sysmaint_group 配置参数 475
sysmon_group 配置参数 476


# T


TCP/IP 服务名称配置参数 474
TEMPSPACE1 表空间 168
territory 配置参数 550
TIMESTAMP 数据类型
  概述 213
Tivoli Storage Manager (TSM)
  管理类配置参数 550
  节点名称配置参数 551
  密码配置参数 552
  所有者名称配置参数 551
tm_database 配置参数 477
toolscat_db 配置参数 560
toolscat_inst 配置参数 561
toolscat_schema 配置参数 561
tp_mon_name 配置参数 477

trackmod 配置参数 550
trust_allclnts 配置参数 479
trust_clntauth 配置参数 479
tsm_mgmtclass 配置参数 550
tsm_nodename 配置参数 551
tsm_owner 配置参数 551
tsm_password 配置参数 552

# U

Unicode
  表和数据注意事项 214
Unicode（UCS-2）
  标识 321
  命名规则 321
UNIQUERULE 列
  查看表的约束定义 259
USERSPACE1 表空间 168
user_exit_status 配置参数 552
util_heap_sz 配置参数 553
util_impact_lim 配置参数
  描述 480

# V

VARCHAR 数据类型
  在表列中 235
varchar2_compat 配置参数
  详细信息 553
vendoropt 配置参数
  详细信息 554
Vista
  用户数据目录 439
Visual Explain
  教程 571
vmo
  AIX 系统命令 5
vmo AIX 系统命令 4

# W

Windows 操作系统
  扩展目录模式
    Windows 2003 89
  Active Directory
    创建 DB2 对象 89
    LDAP 对象类和属性 72
wlm_collect_int 数据库配置参数
  描述 554

# X

XQuery 语句
  不可用 256
XQuery 语句 (续)
  优化
    配置参数 423
  语句堆大小配置参数 549

Printed in China

S151-0612-02

Spine information:

DB2 V9.5 for Linux, UNIX, and Windows V 9 R 5

数据服务器、数据库和数据库对象指南